UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:
DESARROLLO DE UN SISTEMA AUTOMÁTICO DE IRRIGACIÓN PARA HUERTOS CASEROS POR MEDIO DE IOT CON SOFTWARE Y HARDWARE LIBRE

AUTORES:
ESTEFANIA DAYANNA ACOSTA GALLO
CARLOS ROBERTO CUAICAL ANGULO

TUTOR:
JUAN CARLOS DOMÍNGUEZ AYALA

Quito, junio del 2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros Estefania Dayanna Acosta Gallo y Carlos Roberto Cuaical Angulo con documentos de identificación Nº 1726632654 y Nº 1718826421, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación titulado: DESARROLLO DE UN SISTEMA AUTOMÁTICO DE IRRIGACIÓN PARA HUERTOS CASEROS POR MEDIO DE IOT CON SOFTWARE Y HARDWARE LIBRE, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

……

Estefania Dayanna Acosta Gallo

Cédula:1726632654

……..

Carlos Roberto Cuaical Angulo

Cédula:1718826421

Quito, junio del 2021.

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR

Yo declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de Titulación DESARROLLO DE UN SISTEMA AUTOMÁTICO DE IRRIGACIÓN PARA HUERTOS CASEROS POR MEDIO DE IOT CON SOFTWARE Y HARDWARE LIBRE, realizado por Estefania Dayanna Acosta Gallo y Carlos Roberto Cuaical Angulo, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana, para ser considerado como trabajo final de titulación.

Quito, junio del 2021.

Juan Carlos Domínguez Ayala

CI: 1713195590

DEDICATORIA

Este proyecto lo dedico a mi familia que ha estado a mi lado en los buenos y malos momentos, a mis padres Norma Gallo y Francisco Acosta quienes por medio de su trabajo, sacrificio y sabiduría han sido mi inspiración para poder lograr mis metas, a mis hermanas y hermano quienes me han brindado su ayuda, a mis sobrinos Hannah, Amhir, Valentina, Daniel, Emilhy y Mateo quienes han sido una fortaleza para este logro.

Estefania Dayanna Acosta Gallo

Este trabajo lo dedico con mucho cariño a mis padres que son una fuente de inspiración para continuar superándome cada día, a mis hermanas y hermano Thane, Sofia y Alejandro por ayudarme y apoyarme en mi educación, A mis abuelitas Etelvina y Rosa, porque son muy especiales en mi vida y a tía Lili por ayudarme en todo.

Carlos Roberto Cuaical Angulo

AGRADECIMIENTO

Toda mi gratitud va dirigida hacia mis padres Norma Gallo y Francisco Acosta por ser las personas que han trabajado tan duro para educarme todos estos años, muchas gracias por sus enseñanzas y todo su tiempo dedicado a mi formación.

A nuestro tutor Ing. Juan Carlos Domínguez por compartir sus conocimientos con nosotros y por su apoyo brindado en la realización de este proyecto.

Estefania Dayanna Acosta Gallo

Agradezco a mis hermanas por todo su apoyo que me han brindado durante el proceso de formación académica muchas gracias por toda su paciencia y esfuerzo.

Un agradecimiento muy especial a mis padres por apoyarme incondicionalmente en todas las metas que me he planteado.

A todos quienes colaboraron en la elaboración de este proyecto en especial al Ing. Juan Carlos Domínguez.

Carlos Roberto Cuaical Angulo

ÍNDICE

CESIÓN DE DERECHOS DE AUTOR I

DECLARATORIA DE COAUTORÍA DEL DOCENTE TUTOR II

DEDICATORIA III

ÍNDICE V

ÍNDICE DE FIGURAS VII

ABSTRACT XII

INTRODUCCIÓN XIII

CAPÍTULO 1 1

1. ANTECEDENTES 1

1.1. Planteamiento del Problema 1

1.2. Justificación 1

1.3. Objetivos 2

1.3.1. Objetivo General 2

1.3.2. Objetivo Específicos 3

1.4. Marco Conceptual 3

1.4.1. Protocolos de Comunicación en el ámbito de IoT 3

1.4.2. Plataformas de Internet de las cosas y su automatización. 5

1.4.3. Red Sensorial aplicada en IoT 6

1.4.4. Base de Datos en plataformas IoT 7

1.4.5. La nube en plataformas IoT 8

1.5. Estado del Arte basados en IoT 9

1.5.1. Huerto Casero utilizando IoT 9

1.5.2. Sensores usados en huertos caseros 10

1.5.3. Uso de sistemas embebidos en la agricultura 10

1.5.4. Sistemas de riego automático que existen en el mercado 10

CAPÍTULO 2 11

2. PREPARACIÓN Y PLANIFICACIÓN 11

2.1. Infraestructura del Huerto Casero 11
2.2. Sensores que se utilizarán en el sistema automático de riego 12
2.2.1. Sensor de Humedad y Temperatura 12
2.2.2. Sensor de humedad para suelo 13
2.2.3. Sensor de Caudal de agua 14
2.3. Actuadores del sistema automático de riego 14
2.3.1. Electroválvula para la implementación del sistema automático de riego 14
2.3.2. Módulo relé para la activación de electroválvulas 15
2.4. Microcontroladores 15
2.4.1. Raspberry pi 4 16
2.4.2. Esp32 para plataformas IoT 16
2.4.3. Módulo ADS-1115 18
2.5. Protocolo MQTT en plataformas IoT 18
2.6. Cloud de Openhab 18
2.6.1. Openhab 19
2.7. Implementación del Sistema automático de riego para huertos casero. 19
2.7.1. Arquitectura de entradas y salidas del sistema 20
2.7.2. Topología IoT 21
2.7.3. Diagrama de bloques 21
2.7.4. Circuito eléctrico para la irrigación de un huerto casero 26
2.7.5. Diagrama Esquemático del sistema automático de irrigación 27
2.7.6. PCB del Sistema automático de irrigación 28
CAPÍTULO 3 30
3. IMPLEMENTACIÓN DEL PROTOTIPO 30
3.1. Instalaciones en la Raspberry pi 30
3.1.1. Instalación de Openhabian en Raspberry Pi 30
3.1.2. Instalación de Mosquitto MQTT en la Raspberry pi 31
3.1.3. Instalación de la base de datos InfluxDB 31
3.2. Vinculación del proyecto a la nube 32

3.3. Programación de los módulos Esp32 32
3.3.1. Programación del primer módulo Esp32 32
3.3.1.1. Calibración de sensores de humedad capacitivos por el método de gravimetría 34
3.3.2. Programación del segundo módulo Esp32 37
3.3.3. Programación del tercer Esp32 38
3.4. Configuración de un fichero Things en Openhab 38
3.5. Creación del fichero Items 40
3.6. Creación del fichero Persistence 40
3.7. Creación del fichero Sitemaps 41
3.8. Creación del fichero Rules 41
3.9. Creación de una interfaz gráfica mediante Widgets 42
3.10. Configuración de alertas 45
4. ANÁLISIS DEL SISTEMA AUTOMÁTICO DE IRRIGACIÓN 49
4.1. Análisis de Costos 49
4.1.1. Estudio de costo beneficio del sistema de riego IoT 49
4.1.2. Costo de componentes 50
4.1.3. Cálculo del costo 51
4.1.4. Cálculo del beneficio 52
4.1.5. Cálculo de relación costo beneficio 52
4.2. Análisis de Resultados 53
CONCLUSIONES 56
RECOMENDACIONES 58
ANEXOS 63

ÍNDICE DE FIGURAS

Figura 1: Ubicación del huerto implantado. 11
Figura 2: Dimensiones del huerto implantado. 12

Figura 3: Sensor DHT11. 13
Figura 4: Sensor capacitivo de humedad de suelo. 13
Figura 5: Sensor de caudal. 14
Figura 6: Electroválvula. 15
Figura 7: Módulo Relé. 15
Figura 8: Raspberry pi 4. 16
Figura 9: Comparación de módulos ESP. 17
Figura 10: Módulo Esp32. 17
Figura 11: Módulo ADS1115. 18
Figura 12: Interfaz Gráfica de Openhab. 19
Figura 13: Arquitectura del Sistema. 20
Figura 14: Especificación de cada producto. 20
Figura 15: Topología IoT. 21
Figura 16: Diagrama de flujo del Módulo 1. 22
Figura 17: Diagrama de flujo del módulo 2. 23
Figura 18: Diagrama de flujo del módulo 3. 24
Figura 19: Diagrama de flujo del bróker. 25
Figura 20: Circuito Eléctrico de Actuadores y el Sensor de Caudal. 26
Figura 21: Circuito Eléctrico de los sensores capacitivos. 27
Figura 22: Circuito Eléctrico del sensor dht11 y una luminaria. 27
Figura 23: Diagrama Esquemático del Sistema automático de riego. 28
Figura 24: Placa PCB del módulo 1 con los sensores capacitivos 29
Figura 25: Placa PCB del módulo 2 con las electroválvulas. 29
Figura 26: Placa PCB del módulo 3. 29
Figura 27: Configuración de Widget con Gráfica. 43
Figura 28: Gráfica del porcentaje del sensor 1. 43
Figura 29: Configuración del Windget para seleccionar la cantidad de agua que se regara. 44
Figura 30: Selección de cantidad de agua. 44
Figura 31: Interfaz Gráfica en Openhab. 45
Figura 32: Creación de ChatBot en Telegram con BotFather. 46
Figura 33: Configuración de Telegram Bot en Openhab. 46
Figura 34: Condiciones para la activación de la alerta. 47
Figura 35: Script se ejecutará cuando se active la regla. 47
Figura 36: Ejemplo de Alerta recibida al ChatBot. 48
Figura 37: Obtención de Fotografía por medio del ChatBot. 48
Figura 38: Precios de componentes. 50
Figura 39: Precio de mano de obra. 51

Figura 40: Porcentaje de humedad de suelo del sensor 3 sin el funcionamiento del sistema de irrigación automático. 53
Figura 41: Lecturas del sensor de humedad de suelo capacitivo con el sistema de irrigación automático.....54
Figura 42: Porcentaje de humedad de suelo del sensor 2.....54
Figura 43: Ingreso a la interfaz gráfica mediante acceso remoto.....55

ÍNDICE DE ANEXOS

Anexo 1: Presentación de la pantalla de inicio de Openhab. 63
Anexo 2: Configuración con el programa PuTTY.....63
Anexo 3: Configuración para Openhabian.....63
Anexo 4: Herramienta de Openhabian.....63
Anexo 5: Instalación de Mosquitto. 64
Anexo 6: Comando de instalación de base de datos InfluxDB.....64
Anexo 7: Comando para que InfluxDB funcione como servicio. 64
Anexo 8: Comando para iniciar InfluxDB. 64
Anexo 9: Comando de ingreso a fichero de InfluxDB.....64
Anexo 10: Autentificación http.....64
Anexo 11: Configuración de base de datos.....65
Anexo 12: Comando de asignación de permisos. 65
Anexo 13: Configuración de vinculación de Openhab con InfluxDB. 65
Anexo 14: Ingreso de usuario y contraseña. 65
Anexo 15: Selección de base de datos. 66
Anexo 16: Ingreso mediante SSH.....66
Anexo 17: Comando de ingreso al fichero.....66
Anexo 18: Programación del Módulo 1.....67
Anexo 19: Parámetros iniciales para la calibración. 69
Anexo 20: Valores del proceso de calibración.....69
Anexo 21: Gráfica para la función de transferencia del sensor 1.....69
Anexo 22: Gráfica para la función de transferencia del sensor 2.....70
Anexo 23: Gráfica para la función de transferencia del sensor 3.....70
Anexo 24: Lecturas de los sensores calibrados.....70
Anexo 25: Programación del Módulo 2.....71
Anexo 26: Programación del módulo 3 72

Anexo 27: Ingreso al fichero mymqtt.things. 74
Anexo 28: Configuración de Bridge. 74
Anexo 29: Configuración de los Things. 75
Anexo 30: Configuración de canales. 75
Anexo 31: Creación del fichero Huerto.items. 75
Anexo 32: Configuración para enlazar el canal. 75
Anexo 33: Creación del fichero influxdb.persist. 75
Anexo 34: Frecuencia para el almacenamiento de datos en InfluxDB. 76
Anexo 35: Elementos que se van a almacenar en la base de datos. 76
Anexo 36: Creación de Interfaz Gráfica. 76
Anexo 37: Configuración de panel básico. 77
Anexo 38: Interfaz Gráfica en BasicUI. 77
Anexo 39: Comando para fichero rules. 77
Anexo 40: Configuración para ejecutar una regla. 78
Anexo 41: Código YAML. 78
Anexo 42: Configuración de Widget. 78

RESUMEN

En el siguiente proyecto técnico se presenta un sistema de irrigación para huertos caseros que funciona mediante software y hardware libre. Se hace uso de la plataforma de código abierto llamada Openhab, para la automatización del huerto casero, esta permite la creación de una nube privada con la cual el usuario puede interactuar con el sistema de riego desde cualquier parte del mundo.

El sistema integra una arquitectura IoT con tres módulos Esp32 que son programados con el protocolo MQTT y se comunican por medio de una Raspberry pi, la cual será el Bróker en esta implementación, contiene la plataforma Openhab instalada en ella.

El primer módulo Esp32 contiene un sensor de caudal de flujo y es el encargado de activar y desactivar las electroválvulas que permiten el riego del huerto. Por medio de la interfaz gráfica de Openhab el usuario podrá seleccionar la cantidad de agua a regar y los parámetros umbrales de humedad de suelo, todo esto dependerá de las estaciones del año, ya que en invierno va a necesitar menos agua la plantación.

El segundo módulo Esp32 envía los valores de humedad de suelo por medio de los sensores capacitivos de humedad de suelo hacia el Bróker, el cual mediante una regla programada en Openhab, envía la orden de la activación de cada electroválvula.

El tercer módulo Esp32 envía los valores de humedad y temperatura del medio ambiente al bróker y enciende una luminaria para una mejor visualización en condiciones de poca luz dentro del huerto.

ABSTRACT

The following technical project presents an irrigation system for home gardens that works using free software and hardware. The open-source platform called Openhab is used to automate the home garden, this platform allows the creation of a private cloud with which the user can interact with the irrigation system from anywhere in the world.

The system integrates an IoT architecture with three Esp32 modules that are programmed with the MQTT protocol and communicate through a Raspberry pi, which will be the Broker in this implementation, it contains the Openhab platform installed in it.

The first Esp32 module contains a flow sensor and is in charge of activating and deactivating the solenoid valves that allow irrigation of the orchard. Through Openhab graphical interface, the user can select the amount of water to irrigate and the soil moisture threshold parameters, all of this will depend on the seasons of the year, since in winter the plantation will need less water.

The second module Esp32 sends the soil humidity values registered with capacitive soil humidity sensors to the Broker, which by means of a rule programmed in Openhab, sends the order to activate each solenoid valve.

The third module Esp32 sends the values of humidity and temperature of the environment to the Broker and turns on a luminaire for a better visualization in low light conditions inside the orchard.

INTRODUCCIÓN

Este proyecto contiene cuatro capítulos en los cuales se detallará la arquitectura IoT, para la implementación del sistema automático de irrigación en huertos domésticos, en el primer capítulo se encuentra el planteamiento del problema, justificación, los objetivos, el marco conceptual y el estado del arte.

En el capítulo dos se expondrá el lugar en donde se va a implementar el huerto casero, los dispositivos correspondientes para su automatización, la plataforma de software libre con su propia nube, el protocolo y los diseños como la arquitectura, topología, diagramas de flujo, diagrama esquemático, circuito eléctrico y placas PCB.

Para el capítulo tres se ha creado un manual en el cual se explica las distintas instalaciones de programas que se deben realizar en la Raspberry pi, para poder utilizar la plataforma Openhab, posteriormente se realiza programaciones que ayudan con la automatización del huerto casero, se efectúa la calibración de los sensores de humedad de suelo y para finalizar se crea una interfaz gráfica que sea amigable con el usuario.

Finalmente, en el capítulo cuatro se realizó un análisis de costo beneficio, ya que este es el más utilizado en el ámbito de proyectos y el análisis de resultados de la implementación.

CAPÍTULO 1

1. ANTECEDENTES

1.1.Planteamiento del Problema

La mayoría de los sistemas de riego para huertos caseros en el Ecuador no son automatizados, ya que estos tienen costos muy elevados. El abuso de agua es una amenaza creciente para el medio ambiente. Las personas que viven en las grandes ciudades como Quito, apenas tienen tiempo para cuidar de sus jardines. Cuando los cultivos están abandonados durante mucho tiempo, las plantas mueren por exceso de luz solar y por la humedad escasa en las raíces, además el consumo abundante de agua durante el riego puede provocar una oxigenación incorrecta de las plantas. Por tanto, es necesario automatizar el proceso de riego para asegurar que las plantas reciban suficiente agua. ¿Es posible realizar un sistema de riego para huertos caseros que funcione en base a IoT (Internet Of Things), usando software y hardware libre?

1.2.Justificación

La automatización de este sistema aprovecha los recursos hídricos para poder mejorar los cultivos, el uso de un riego correcto en las plantaciones disminuye el impacto ambiental, el cual es causado por parte del mal uso del agua. Para ello la automatización realizada podrá recolectar datos en tiempo real de la humedad y por medio de algoritmos determinados se podrá representar en una interfaz de forma remota por medio de un servidor.(Mecánica, por, and Diego Cruz Freire Darwin Vinicio Chimbo Chimbo 2015)

Los huertos domésticos predestinan todos sus productos para su propio consumo y a la comercialización creando una gran rentabilidad, ya que existe una demanda de mercado al producto alimentario orgánico. La combinación de la tecnología con la agricultura ha logrado crear la noción de una agricultura precisa, por lo que la gestión terrenal mediante la monitorización de variables que existen en las plantaciones, así como el procesamiento y la actuación de estas, pueden llegar a obtener un sistema de riego con un buen funcionamiento. Así mismo, el cultivo para las familias terratenientes son un ingreso económico, por lo que mercantilizan una parte del producto obteniendo un capital extra, por otra parte, se asegura de que el producto

tenga una seguridad alimentaria siendo un aporte esencial para la salud del consumidor mediante la buena alimentación. (Selmani et al. 2018)

Por medio de la automatización del sistema de riego se logrará obtener un cultivo orgánico de gran calidad, ya que, gracias a este las plantaciones van a recibir el agua con los nutrientes necesarios, para que el cultivo se pueda reproducir de una manera eficaz, haciendo que el huerto siempre se mantenga en buen estado hasta el día de la cosecha.

En este proyecto se adquirirá conocimientos sobre el manejo del protocolo MQTT (Message Queue Telemetry Transport) en sistemas embebidos, para poder conectar los sensores hacia el protocolo es necesario realizar una red inalámbrica de sensores, los cuales conformarán un nodo sensor de la red que se conectarán a un nodo central, el cual será el encargado del envió de los datos hacia la plataforma IoT. El protocolo MQTT cuando trabaja con sensores y actuadores tiene un comportamiento fácil de manipular ya que este protocolo fue creado con la finalidad de realizar intercambios de información entre dispositivos con un bajo ancho de banda y consumo energético.(Villena et al. 2019)

IoT tiene una plataforma la cual guarda datos para estudios futuros y para ello se necesita un base de datos que cumpla con las necesidades que se requiera en la implementación de la plataforma para que sea eficiente y segura, existen varias bases como SQlite, Orancle, InfluxDB, MySQL (My Structured Query Language) entre otras, todas estas bases de datos se las puede vincular a Node-Red, Openhan, Home Assitant, estas aplicaciones funcionan con Grafana para la personalización de gráficas.(Nițulescu and Korodi 2020)

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar un sistema automático de riego para huertos caseros con IoT usando

sistemas embebidos, que determine la irrigación necesaria en el huerto.

1.3.2. Objetivo Específicos

- Analizar el estado del arte de sistemas automáticos de Irrigación a través de IoT para recopilación de estudios previos sobre el tema.

- Diseñar la propuesta de un sistema de riego automático en un huerto casero específico para la determinación de los componentes que formarán parte del sistema automatizado.

- Implementar un prototipo con funciones de un sistema automático de riego basado en IoT, con la obtención de las variables para la configuración del Sistema.

- Analizar los costos de la red IoT de irrigación de cultivos para la validación de la factibilidad de una futura implementación.

1.4. Marco Conceptual

1.4.1. Protocolos de Comunicación en el ámbito de IoT

El Internet de las cosas (IoT) tiene varias plataformas, en las cuales es necesario la incorporación de protocolos para su funcionamiento, ya que la comunicación de estas tecnologías funciona por medio de M2M (Machine to Machine). Se debe tomar en cuenta las necesidades del usuario y de su implementación, para la elección correcta del protocolo, a continuación, se realizará una breve descripción de cada protocolo para plataformas de IoT.(Ramirez and Pedraza 2017)

- MQTT (Message Queue Telemetry Transport): Este protocolo tiene la finalidad

de interconectarse a mecanismos embebidos y computadoras utilizando aplicaciones y equipos de mensajería, este protocolo se maneja por medio de publicaciones, suscriptores e intermediarios, haciendo que el bróker avise a los subscriptores que se ha publicado ciertos topics, los cuales pueden ser de su agrado y puedan subscribirse a estos, después de que existan subscriptores, el Publisher comunicará al bróker que existen nuevas publicaciones y este entregue la información a los subscriptores sobre los topics requeridos, este protocolo tiene incorporado 3 tipos de QoS (Quality of service), que representa la calidad de servicio con la que se están transmitiendo los mensajes, tomando en cuenta que el QoS 0 solo verifica que el mensaje solicitado haya sido enviado; QoS 1 verifica que el mensaje haya llegado al destino sin complicaciones sin importar la cantidad de mensajes enviados; el QoS 2 verifica que todos los mensajes hayan sido entregados sin ninguna complicación en el camino, esta calidad de servicio tiene sobrecarga de mensajes a comparación con la QoS 1 que es más veloz por lo que esta es la más utilizada, se puede decir que gracias al QoS el protocolo MQTT es excelente en mensajería, por lo que es eficiente para las plataformas de IoT con dispositivos de bajo costo.(Ramirez and Pedraza 2017)

- Constrained Application Protocol (CoAP): Utiliza el protocolo UDP (User Datagram Protocol) y manipula ciertas funciones de HTTP (Hypertext Transfer Protocol), para su funcionamiento en plataformas IoT, este protocolo pertenece a la capa de aplicación, por este motivo se puede utilizar en IoT, tiene un Software para equipos electrónicos minúsculos, que sean dinámicos en el ciberespacio y precisen ser dominados en un sitio apartado, utilizando sistemas que usen el protocolo HTTP y requieran enlazarse a una interfaz.(Ramirez and Pedraza 2017)

- Extensible Messaging and Presence Protocol (XMPP): Este protocolo se usa para que las personas puedan comunicarse por medio de internet con chats y videollamadas, a través de cualquier sistema operativo que ofrezca un buen régimen de seguridad y que tenga una afinidad con los protocolos, ya que utiliza

lenguaje extensible de marcado con estructuras de registro, anuncio y contestación que funcionan con datos en tiempo real.(Ramirez and Pedraza 2017)

- Data Distribution Service (DDS): Este protocolo se asemeja a MQTT porque funciona con publicaciones, subscripciones y con recolección de datos en tiempo real, lo que este protocolo no tiene es Bróker, aunque trabaja con varias redes para poder tener una buena calidad de servicio y así lograr entrelazar a las plataformas IoT.(Ramirez and Pedraza 2017)

1.4.2. Plataformas de Internet de las cosas y su automatización.

Internet de las cosas tiene una plataforma que facilita que todos los usuarios se puedan conectar desde cualquier dispositivo electrónico sin importar su sistema operativo con la condición de que tenga acceso a internet, para que se pueda conectar en cualquier ubicación e instante a la plataforma IoT y pueda la persona verificar cómo va el funcionamiento de su implementación.(Puranik et al. 2019)

Esta plataforma funciona con varios dispositivos pequeños que tengan un consumo energético limitado y necesiten ser monitoreados en un área determinada, para que esta aplicación sea eficiente consta de algunos parámetros como:

- Tener la capacidad suficiente para sustituir el trabajo humano por medio de la automatización realizada a los dispositivos.

- Por la utilización de protocolos tiene una calidad de servicio relevante y una excelente comunicación entre ellos.

- Por medio de la información recolectada en la base de datos se realizan estudios a futuro. (Nageswara Rao and Sridhar 2018)

Las Plataformas IoT en el presente ejecuta aplicaciones que tengan implementaciones con redes de sensores y middleware el cual hace que exista una buena comunicación entre aplicaciones y protocolos para poder acceder a las plataformas sin ningún inconveniente. (Shilpa, Muneeswaran, and Devi Kala Rathinam 2019)

1.4.3. Red Sensorial aplicada en IoT

Esta red está conformada por una variedad de sensores que se encuentran ubicados en una determinada implementación, ya sea en un campo abierto o cerrado, estos sensores se deben empalmar a los nodos que forman parte de la red de sensores, los nodos sirven para reunir y manipular información almacenada por los sensores en un área determinada, estos reunirán información sobre algunos factores del medio ambiente, como la temperatura, humedad ya sea del aire o del suelo, si existe fluido se podría tomar valores del PH, caudal, entre otros.(Mir and Khachane 2018)

Por otra parte, esta red de nodos consta de un sistema eléctrico interno que está compuesto por procesadores de nodos, dispositivos para una buena comunicación entre sensores y finalmente un recurso energético para cada nodo de la red de sensores, ya que su funcionamiento es de forma independiente, la dimensión de estos nodos es pequeña por lo que tiene una capacidad de memoria corta.(Mir and Khachane 2018)

La red de sensores se puede implementar en algunos ámbitos como en casas domóticas, hospitales, huertos caseros y fabricas industriales, puesto que en estos lugares se necesita tener un monitoreo constante de las actividades y a través del Internet de las cosas facilita la automatización de los equipos implementados. (Pratyush Reddy et al. 2020)

Los sensores que son parte de la Red sensorial siempre están en constante obtención de datos y acumulación de estos, en su unidad de almacenamiento, toda la información recolectada va a ser enviado solo si acoge una petición de la estación de base o si pasa algún acontecimiento. (Pratyush Reddy et al. 2020)

1.4.4. Base de Datos en plataformas IoT

Estas bases de datos recolectan y guardan información que envían los sensores, con la finalidad de que la información recopilada sirva para análisis futuros y se utiliza para distintas aplicaciones como sistemas de monitoreo meteorológicos, ganadería, huertos caseros, historias clínicas, insumos médicos, entre otras, a continuación, se realizará una rápida descripción de estas.(Di Martino et al. 2019)

- Apache Cassandra: Es un sistema de control de base de datos que no trabaja de maestro a esclavo, ya que recolecta datos como una tabla, es decir de forma vertical, tiene un código fuente estructurado conocido como Cassandra Query Languaje y sus datos recopilados son secuenciales y en tiempo real.(Di Martino et al. 2019)

- MongoDB: Es una base de datos en tiempo real, con programación orientada a objetos que utiliza la extensión JSON, que opera con pares clave-valor, estas trabajan con claves exclusivas, facilitando la mejora de la obtención de información que se distribuirá de manera uniforme en los nodos.(Di Martino et al. 2019)

- InfluxDB: Esta data base se maneja con datos en tiempo real, los cuales serán organizados de acuerdo con el periodo en el que se haya recolectado esa información, posee una política de retención de registros con datos obsoletos recopilados, también tiene asesoramiento constante y guarda los datos en un sistema de administración de base de datos, conforme a esto se utiliza un sistema de recolección de registro mixto por ello la información se ordenará en serie o columna.(Di Martino et al. 2019)

- SQLite: Es una base de datos que se puede conectar con diversos dispositivos, que tienen sistemas operativos como Linux, Windows, macOS y que no necesita un servidor para funcionar o de conexiones extrínsecas, esta base de datos tiene una capacidad de almacenamiento hasta un terabyte, es un software libre y su funcionamiento es mediante API (Application Programming Interface) que son muy simples de ocupar, ya que tienen una guía de intervenciones con su descripción correspondiente para facilitar su uso. (Widyawati, Ambarwari, and Wahyudi 2020)

1.4.5. La nube en plataformas IoT

Una nube no tiene un almacenamiento local, esta se maneja por medio de archivo de datos en el internet, en el cual se puede guardar cualquier tipo de información en ella desde imágenes, documento hasta programas sin la necesidad de ocupar la memoria que proporciona el computador, ya que este se opera en el ciberespacio. Consta de tres tipos de nubes que se detallaran de una manera rápida a continuación. (Avşar et al. 2018)

- Nube Pública: La seguridad es un factor muy importante en lo que se refiere en el ámbito de la red, pero esta nube tiene esa gran problemática, ya que esta no consta de una buena seguridad, por lo que pierde confianza al momento de ser utilizada, por este gran motivo las grandes empresas no contratarían un servicio como este, porque cualquier sujeto ingresaría a la nube y violaría la privacidad de ella, utilizando los archivos de la Compañía en beneficio propio. (Avşar et al. 2018)

- Nube Privada: Al contrario de la nube publica, esta si tiene una firmeza en la privacidad, es decir que la empresa puede tener intimidad sin ningún miedo a ser hackeados, por ello también utilizan una VPN (Virtual Private Network) para que se vuelva más segura y sus archivos se encuentren protegidos. (Avşar

et al. 2018)

- Nube Híbrida: Es la unión entre la nube privada y la pública, colocando una VPN privada adentro de una pública, obteniendo eficacia y confianza a la vez, esto la ha convertido en la nube más solicitada en el ámbito de las Compañías y por ello las empresas contratan este servicio híbrido. (Fujii and Koike 2017)

1.5. Estado del Arte basados en IoT

Se ha efectuado una investigación acerca de sistemas automatizados en huertos caseros, que usan plataformas IoT, para la realización de estos se ha tenido que examinar ciertos parámetros como: sensores, variables que se encuentran en el medio ambiente, sistemas embebidos y sistemas que existen en el mercado.

Algunas soluciones que se da para sistemas embebidos con plataformas IoT, ofrecen un sistema de telecontrol, en el cual el usuario podrá interactuar por medio de estas, algunas plataformas ofrecen recolectar y guardar información en una nube privada.

1.5.1. Huerto Casero utilizando IoT

Es un lugar dónde se realizará la siembra de varios productos alimenticios orgánicos, que se ubicarán en balcones, terrazas, patios, entre otros. Estos lugares deben tener la luz necesaria para el desarrollo de las plantas, ya que si no tienen la temperatura necesaria las plantas pueden enfermar a tal punto que llegarían a la muerte, pero no solo es asunto de la temperatura sino también del agua, porque necesitan de ella para su reproducción, debido a que esta transporta los nutrientes necesario para el desarrollo de una buena plantación, hay que tomar en cuenta la temporada de siembra y cosecha de cada tipo de planta, a causa de esto se orientará este proyecto al sistema de automatización de irrigación. (Assaf and Ishaq 2020)

1.5.2. Sensores usados en huertos caseros

Dentro de un huerto casero los sensores son los encargados de la recopilación de información de las variables del medio ambiente, las cuales determinan el desarrollo correcto de las plantas, las variables tanto de la humedad del suelo, temperatura en el huerto, luz solar y el nivel de agua almacenado en el tanque de riego. Todas estas se pueden adquirir haciendo uso de una red de sensores, ya que estos elementos son esenciales para lograr un cultivo preciso. (Veloo et al. 2019)

Los sensores capacitivos de humedad de suelo son de bajo costo y tienen una alta precisión, el método recomendado para su calibración es por medio de gravimetría. (Radi et al. 2018)

1.5.3. Uso de sistemas embebidos en la agricultura

La digitalización de cultivos es cada día más popular, debido a que gracias a esto los usuarios pueden monitorear los cultivos por medio de dispositivos móviles. Los sistemas embebidos tienen un rol importante en la automatización de los huertos, porque son los encargados del cumplimiento y gestión de las operaciones que se deben realizar para el riego, los elementos que contienen estos sistemas son baratos y su consumo de energía es baja. (Puranik et al. 2019)

1.5.4. Sistemas de riego automático que existen en el mercado

Hay una variedad de sistemas autónomos de riego que ofrecen diferentes formas para controlar su funcionamiento como, por ejemplo: sistemas controlados con temporizadores, PLC (Programmable Logic Controller), entre otros, los cuales existen hoy en día, pero estos sistemas son costosos. Los sistemas basados en IoT permiten obtener huertos con excelentes rendimientos, para lograr un huerto casero de precisión, por lo que la implementación de estos sistemas de riego es muy importante. Estos sistemas son populares en plantaciones, ya que tienen algunas ventajas como la obtención de huertos saludables, tener una mayor producción y el uso eficiente del agua, gracias a esto en los hogares se tendrá una alimentación sana con productos orgánicos. (Lafuente 2019)

CAPÍTULO 2

2. PREPARACIÓN Y PLANIFICACIÓN

2.1.Infraestructura del Huerto Casero

Este huerto doméstico se implementará en Sangolquí a la altura del Club los Chillos como se muestra en la siguiente Figura 1, tiene una altitud de 2.550 metros m.s.n.m (metros sobre el nivel del agua), superficie de 10,25 m^2 y temperatura promedio ambiental es de 14,1°C, con una humedad promedio del 84%. (climate-data.org, s.f.)

Figura 1: Ubicación del huerto implantado.

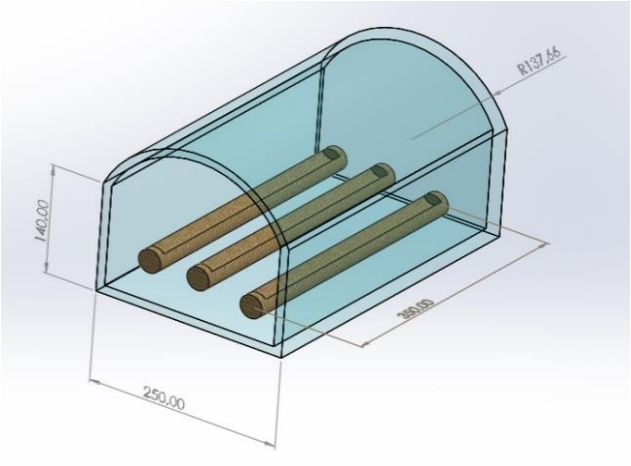
Ubicación de huerto implantado en el cantón Sangolquí Fuente: Google Earth.

Para la ejecución del sistema automático de riego se eligió un lugar amplio donde se tomó en cuenta que el huerto pueda obtener la mayor cantidad de luz solar, determinado con un luxómetro que en esta área existe 179.489 lux en un día despejado y en un día nublado 2.509 lux, por este motivo se consideró crear un pequeño invernadero en esta área.

Para la creación del huerto se utilizará tubería y plástico de invernadero reciclado, el

cual consta de las siguientes medidas: 4.10m de largo, 2.50 m de ancho, 2.20m de altura. Después de llevar a cabo la construcción de este campo como se muestra en la Figura 2, se realizará el proceso de siembra.

Figura 2: Dimensiones del huerto implantado.



Elaborado por: Estefania Acosta & Carlos Cuaical.

2.2.Sensores que se utilizarán en el sistema automático de riego

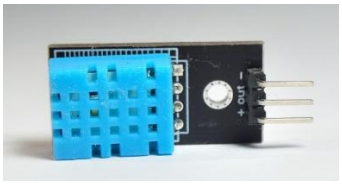
Para la determinación de estos, se debe tener presente las variables que se van a tomar en cuenta para automatizar el sistema de irrigación, para ello se necesita controlar temperatura y humedad que son importantes para el control de riego, por esta razón se debe tomar en cuenta el medio ambiente y el suelo, además del caudal de agua, para la elección de estos dispositivos se realizará comparaciones que demuestren que el instrumento seleccionado sea el más eficiente para la automatización.

2.2.1. Sensor de Humedad y Temperatura

Estos constan de un termistor que detecta la temperatura del entorno y un sensor capacitivo que sirve para captar la humedad del medio ambiente, para esta implementación se ha seleccionado el sensor DHT11, el cual tiene incluido una resistencia, su recubrimiento es de color azul, cuyas dimensiones son de largo 16 mm, ancho 12.45mm y profundidad 5.56mm, consta de 3 pines de salida como se muestra

en la Figura 3, se puede alimentar con un voltaje de salida de 3.3V a 5V, siendo su salida una señal digital, con una escala de temperatura de sensado que va desde 0°C a 50°C, con una exactitud de + - 2°C, así como la captación de humedad relativa va desde un valor de 20% hasta 90% y para finalizar su periodo de contestación es de 1s. (Margret Sharmila et al. 2019)

Figura 3: Sensor DHT11.



Elaborado por: Estefania Acosta & Carlos Cuaical.

2.2.2. Sensor de humedad para suelo

Existen 2 tipos de sensores que se pueden implementar para este proyecto y estos son los sensores capacitivos y resistivos, ya que son de fácil adquisición. El detector FC32 es el más económico, pero este tiene las pistas de pcb expuestas, esto ocasiona que se degrade por la humedad y provoque contaminación tanto en el suelo como en la planta, porque su oxidación libera químicos perjudiciales para la siembra, el SKU SEN0193 como se muestra en la Figura 4,es costoso, no tiene ninguna parte metálica exhibida, por ende es más duradero para los ambientes húmedos y por este fundamento se lo ha elegido para esta implementación, además trabaja con un voltaje de 3.3V a 5V y es ideal para aplicaciones de bajo consumo energético. (Radi et al. 2018)

Figura 4: Sensor capacitivo de humedad de suelo.

Elaborado por: Estefania Acosta & Carlos Cuaical.

2.2.3. Sensor de Caudal de agua

Con la ayuda del sensor de caudal como se muestra en la Figura 5, se puede medir el paso de agua y gracias a este se puede saber la cantidad de fluido, la cual será entregada al huerto casero, de esta manera se obtiene el consumo de agua, dentro del captador de flujo tiene implementado una rueda constituida por un imán y un sensor hall que detecta el cruce del dispositivo magnético, haciendo que el instrumento envié los pulsos y que el microcontrolador sea el encargado de traducir este conteo a gasto de líquido o a la velocidad que se maneja este.

Figura 5: Sensor de caudal.

Elaborado por: Estefania Acosta & Carlos Cuaical.

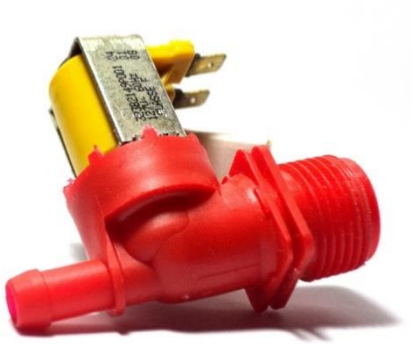
2.3.Actuadores del sistema automático de riego

Los actuadores son esenciales para el control de sistemas automatizados, ya que ellos son los encargados de cumplir un grupo de instrucciones que son determinadas por el controlador, a continuación, se detallará algunos dispositivos que serán utilizados en este proyecto.

2.3.1. Electroválvula para la implementación del sistema automático de riego

Es un elemento electromecánico como se muestra a continuación en la Figura 6, tiene como función autorizar o inmovilizar el paso del líquido a través de un conducto, su manejo es por medio de un electroimán el cual genera un campo electromagnético cuando recibe electricidad.

Figura 6: Electroválvula.



Elaborado por: Estefania Acosta & Carlos Cuaical.

2.3.2. Módulo relé para la activación de electroválvulas

Este dispositivo como se muestra en la Figura 7, está constituido internamente por una bobina de cobre, la cual genera un campo magnético y acciona un interruptor, al momento que ocurre esto, se permite el paso de voltajes altos, es importante proteger el microcontrolador de cargas altas, la selección de estos se realiza en función de voltaje y corriente con el que se desee operar, el módulo JQC 3FF SZ es el que se utilizará en este proyecto, para encender las electroválvulas.

Figura 7: Módulo Relé.

Elaborado por: Estefania Acosta & Carlos Cuaical.

2.4.Microcontroladores

Estos instrumentos están constituidos por memoria, procesador y dispositivos

periféricos, pueden ser programados con IDE de Arduino, MicroPhyton, entre otros. Algunos mecanismos como la Raspberry pi se puede programar mediante comandos de Linux o Windows dependiendo del sistema operativos que se seleccione. Los microcontroladores fueron fabricados para ordenadores con aplicaciones de bajos recursos.

2.4.1. Raspberry pi 4

Existen varias topologías con distintos protocolos a nivel mundial, pero la que se empleará en este proyecto es la que utiliza la arquitectura del protocolo MQTT, para ello es necesario tener un servidor, por esto se utilizará la Raspberry pi, este componente se lo puede observar en la siguiente Figura 8, la cual será el Bróker en este diseño, ya que es el encargado de avisar los topics publicados a los subscriptores.

Figura 8: Raspberry pi 4.

Elaborado por: Estefania Acosta & Carlos Cuaical.

2.4.2. Esp32 para plataformas IoT

Para la elección de este módulo se realizó una comparación entre el Esp8266 y el Esp32 como se muestra en el Figura 9

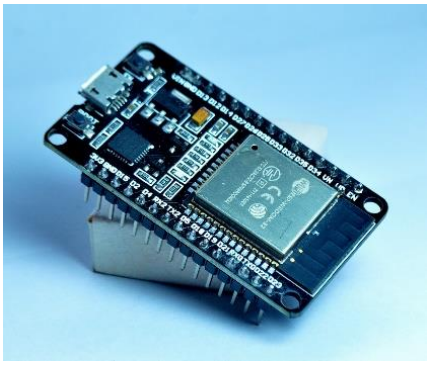
Figura 9: Comparación de módulos ESP.

COMPARACIÓN DE MÓDULOS ESP		
CARACTERÍSTICAS	**ESP32**	**ESP8266**
Wifi	802,11n -150Mbps	72,2 Mbps
GPIO	36	17
Bluetooth (Low energy)	1	0
ADC	8	1
Sensor de Temperatura	1	0
Sensor Hall	1	0
Sensor Táctil	8	0
Protocolos	TCP, UDP, HTTP, FTP, MQTT, IPV4, IPV6, SSL	TCP, UDP, HTTP, MQTT, IPV4
SRAM	520 KB	< 50 KB
ROM	448 KB	0
Elección para el proyecto	X	

Comparación y elección del módulo ESP, Elaborado por: Estefania Acosta & Carlos Cuaical

En esta figura se puede observar que la mejor elección es el dispositivo Esp32, ya que su funcionamiento es mediante Wifi y Bluetooth LE (Low Energy), por lo que este dispositivo se puede utilizar con una variedad de plataformas IoT, ya que se adapta a algunos lenguajes de programación con facilidad, el uso de la red inalámbrica es de buena calidad, así como la conexión bluetooth es de poco consumo, posee una gran cantidad de puertos ADC para la implementación de sensores análogos en la Figura 10 se puede ver la apariencia física de este módulo.

Figura 10: Módulo Esp32.

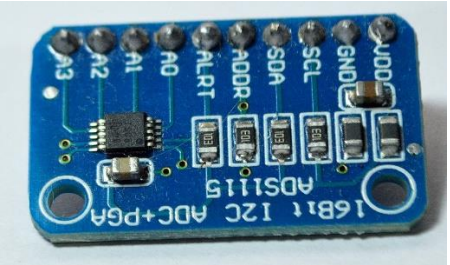


Elaborado por: Estefania Acosta & Carlos Cuaical.

2.4.3. Módulo ADS-1115

Este conversor Análogo – Digital tiene 16 bits, 4 entradas análogas, en las que se pueden conectar varios sensores, consta de un amplificador de ganancia programable (PGA) por lo que se puede obtener lecturas más precisas, funciona con comunicación I2C y un comparador digital la apariencia física de este módulo se la puede ver en la Figura 11. (Choudhury, Moulik, and Choudhury 2020)

Figura 11: Módulo ADS1115.



Elaborado por: Estefania Acosta & Carlos Cuaical.

2.5.Protocolo MQTT en plataformas IoT

Este protocolo se puede asociar con dispositivos embebidos por medio de aplicaciones y mensajes de texto por ello, es eficiente trabajar con plataformas IoT, MQTT se maneja por medio de publicaciones, suscripciones y Bróker el cual será la Raspberry pi en este proyecto, su comunicación es a través de topics y tiene 3 niveles de calidad de servicio los cuales pueden ser ocupados dependiendo de lo que se necesite realizar en la implementación.

2.6.Cloud de Openhab

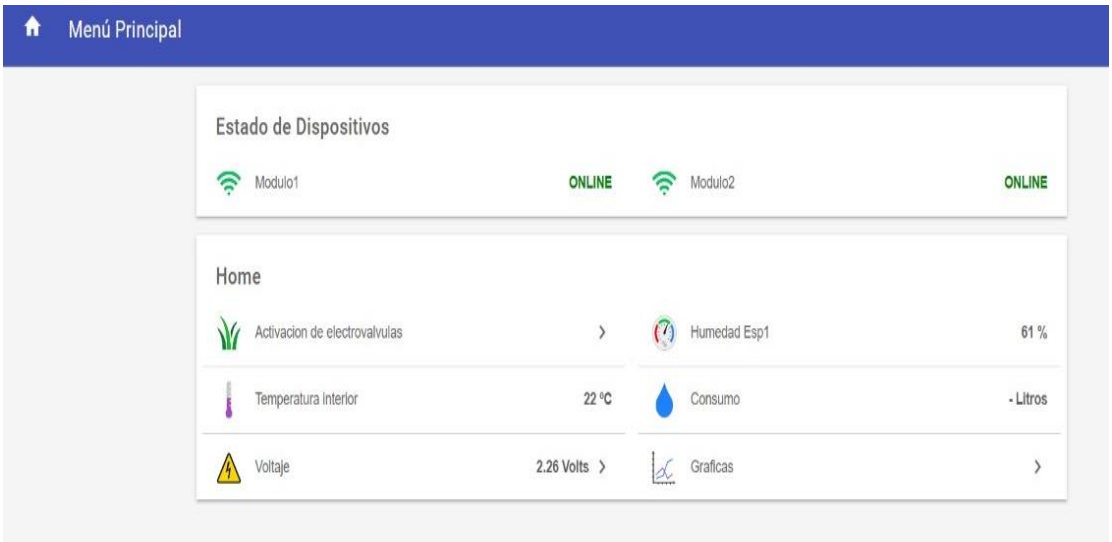
Openhab consta de una nube propia la cual permite el almacenamiento de datos, tiene entrada segura ya que toda la información solicitada se maneja a través de un túnel, también consta de un repositorio de persistencia, alertas de texto y un Back-end que sirve para asegurar que todo funcione de manera adecuada en el Cloud, todos los datos recolectados en esta nube se los puede visualizar mediante la interfaz gráfica, puesto

que pueden servir para estudios futuros y para el control de las variables implicadas en el sistema de irrigación automático.

2.6.1. Openhab

Es un sistema de automatización de código abierto para viviendas, que permite conectar varios dispositivos y proveer la información en una interfaz gráfica como se muestra en la Figura 12, es capaz de programar alertas, eventos, entre otros. Este puede ser implementado en varios sistemas operativos como Linux, Mac Os, Windows y Raspberry pi, consta de su propia nube para la recolección de datos y se lo puede manipular por cualquier dispositivo electrónico que tenga acceso a Internet.

Figura 12: Interfaz Gráfica de Openhab.



Elaborado por: Estefania Acosta & Carlos Cuaical

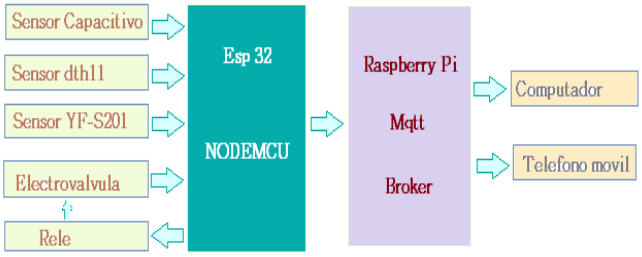
2.7.Implementación del Sistema automático de riego para huertos casero.

Después de haber determinado todos los dispositivos para este proyecto, a continuación, se detallará la arquitectura del circuito, la topología IoT, circuitería y algunos diagramas necesarios para la implementación de este, para ello se ha utilizado algunos programas como Lucid para los diagramas y para el diseño electrónico se empleó EasyEDA.

2.7.1. Arquitectura de entradas y salidas del sistema

Como se puede visualizar en la Figura 13, la arquitectura consta de varios sensores que realizan la toma de medidas del medio ambiente, suelo, agua que serán enviados al Esp32, esta información es remitida por medio del protocolo MQTT hacia el Bróker (Raspberry pi), el cual recibe los valores y los almacena en la base de datos InfluxDB, estos se pueden visualizar a través de gráficos e iconos programados en Openhab.

Figura 13: Arquitectura del Sistema.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Para una mejor compresión de que función cumplirá cada dispositivo en este proyecto se puede observar en la Figura 14.

Figura 14: Especificación de cada producto.

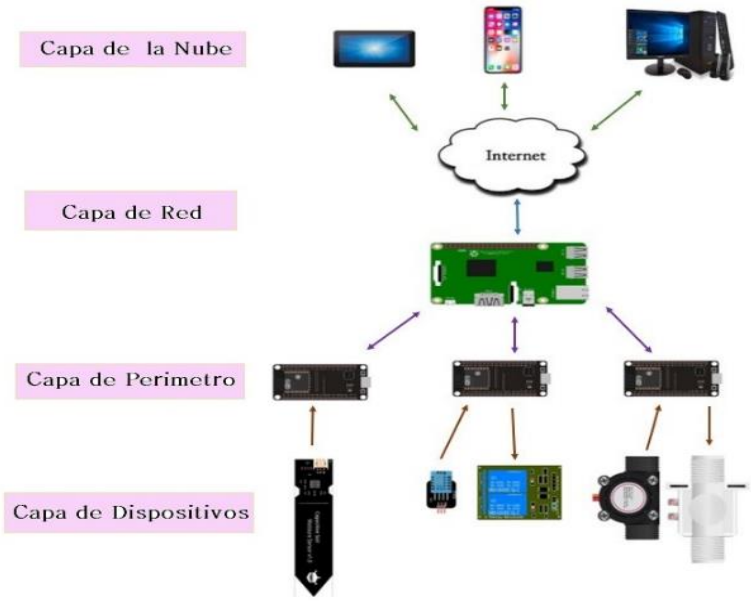
COMPONENTES	MEDICIÓN	ESPECIFICACIONES
Sensor Capacitivo	Humedad de Suelo	Activación de Electroválculas Desactivación de Electroválvulas
Sensor dht11	Temperatura y Humedad del medio ambiente	Envía valores tomados del medio ambiente, cada minuto al Bróker
Sensor YF-S201	Caudal del agua	Envía valores de consumo de agua en litros y cada día se reinicia para la toma de nuevas medidas.
Relé	/	Sirve para la activación de Electroválvulas y encender la luz del invernadero.
Esp 32	/	Envía y recibe información del Bróker (Raspberry pi).
Raspberry Pi	/	Procesa información para ejecutar órdenes y almacena en la base de datos los valores recibidos.
Openhab	/	Interfaz Gráfica
Protocolo Mqtt	/	Es el encargado de la conexión de los dispositivos que funcionan con el estándar 802.11

Especificación de las tareas de cada dispositivo, Elaborado por: Estefania Acosta & Carlos Cuaical.

2.7.2. Topología IoT

El protocolo MQTT tiene incorporado la topología en forma de estrella como se muestra en la Figura 15, es decir que el Bróker se encuentra en el núcleo y es el encargado de publicar mensajes, los clientes se subscriben a publicaciones enviadas por el nodo central. La capa de perímetro está compuesta por controladores que recolectan información desde la capa de dispositivos, los cuales van a conectarse con la capa de red, para poder realizar un análisis con los datos obtenidos y tomar ciertas acciones con el fin de tener control en el sistema automatizado, por medio de la resolución y las alarmas implementas, estas se pueden visualizar mediante la capa de nube, ya que consta con artefactos con interfaz gráfica.

Figura 15: Topología IoT.



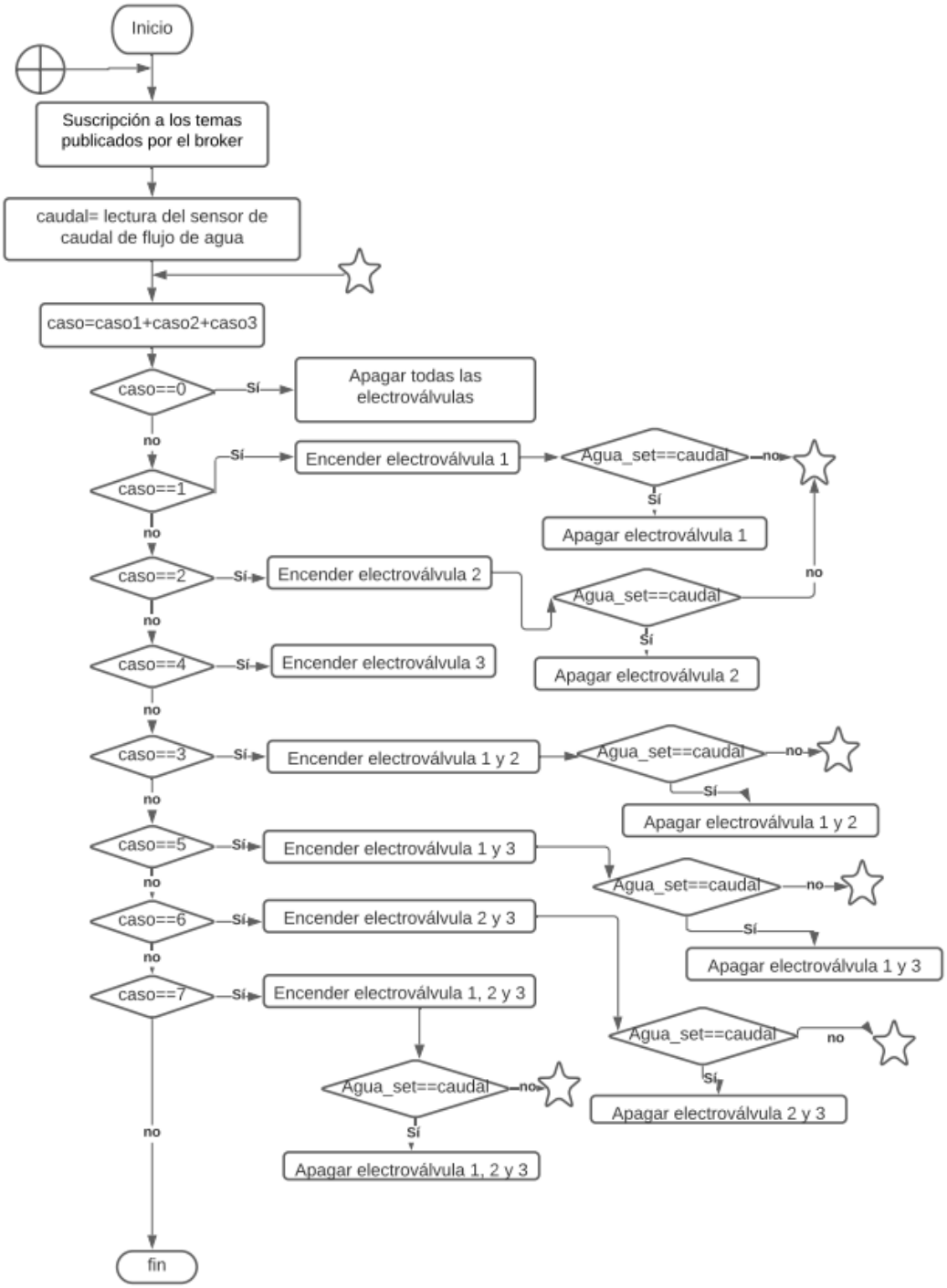
Elaborado por: Estefania Acosta & Carlos Cuaical.

2.7.3. Diagrama de bloques

A continuación, se muestra en la Figura 16, el diagrama de flujo modulo 1, que es el encargado de activar las electroválvulas cuando recibe los mensajes del bróker con los números 1,2 y 4. Estos números representan los casos son almacenados en una variable que se encarga de sumarlos para determinar mediante una estructura de control switch case la electroválvula o las electroválvulas que deben ser activadas. Una vez que se ejecuta un caso, la o las electroválvulas no se desactivan hasta que el sensor de flujo

detecte que el valor de agua regada es igual al valor umbral de agua que el usuario configura en la interfaz gráfica de OpenHAB.

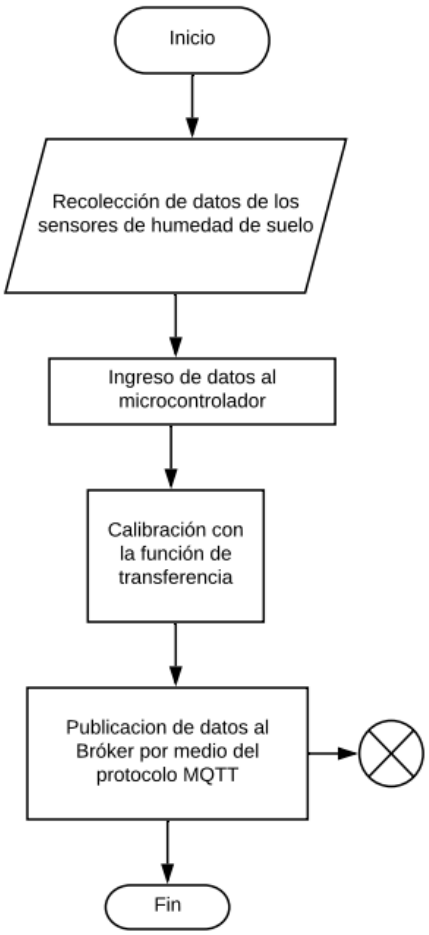
Figura 16: Diagrama de flujo del Módulo 1.



Elaborado por: Estefania Acosta & Carlos Cuaical.

El diagrama de bloques de la Figura 17 representa la configuración del módulo 2 que se encarga de recolectar los valores detectados por los sensores de humedad capacitivos, los valores ingresan a una función de transferencia para ser calibrados y luego se envían por medio de una publicación hacia el bróker.

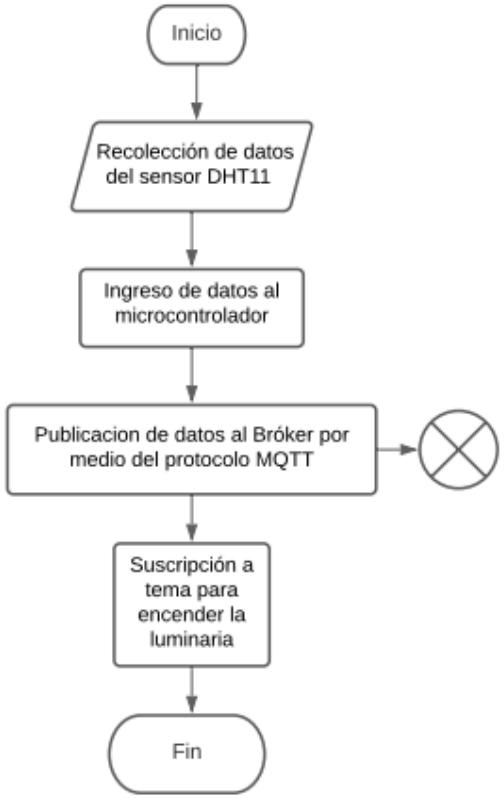
Figura 17: Diagrama de flujo del módulo 2.



Elaborado por: Estefania Acosta & Carlos Cuaical.

El módulo 3 se encarga de recolectar los valores de humedad y temperatura del medio ambiente, estos valores son publicados hacia el bróker, este módulo también se suscribe a un tema que envía el bróker para encender una luminaria, este proceso está representado en el diagrama de bloques de la Figura 18.

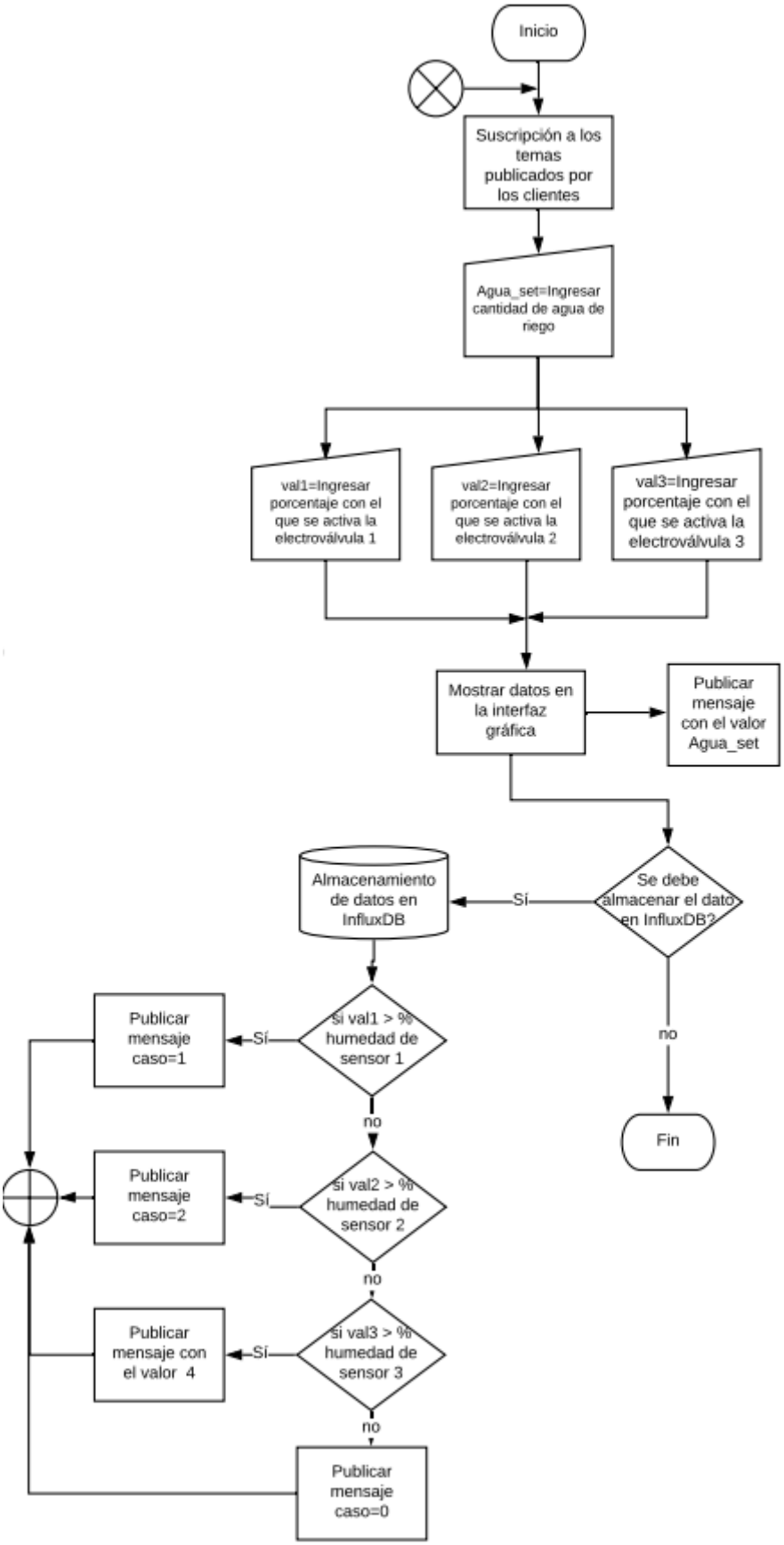
Figura 18: Diagrama de flujo del módulo 3.



Elaborado por: Estefania Acosta & Carlos Cuaical.

En la Figura 19 se muestra el diagrama de flujo del bróker, el cual se suscribe a todos los mensajes que envían los módulos Esp32 y también recibe las configuraciones que el usuario establece como parámetros umbrales para el riego automático, estos parámetros son: la cantidad de agua que se desea regar y también el porcentaje de humedad de suelo con el cual se desea efectuar la irrigación, como son tres sensores de humedad capacitivos se configuran tres parámetros de umbrales de porcentaje de humedad de suelo, estos valores umbrales son comparados con el porcentaje de humedad actual y el porcentaje de humedad actual es inferior al valor umbral configurado se publica un mensaje con la respectiva electroválvula que debe activarse. Los datos que se obtienen de los módulos son visualizados desde la interfaz gráfica de Openhab, los datos de humedad, temperatura y porcentaje de humedad son almacenados en la base de datos InfluxDB, mientras que los datos de encender y apagar la luminaria no son almacenados en la base de datos.

Figura 19: Diagrama de flujo del bróker.

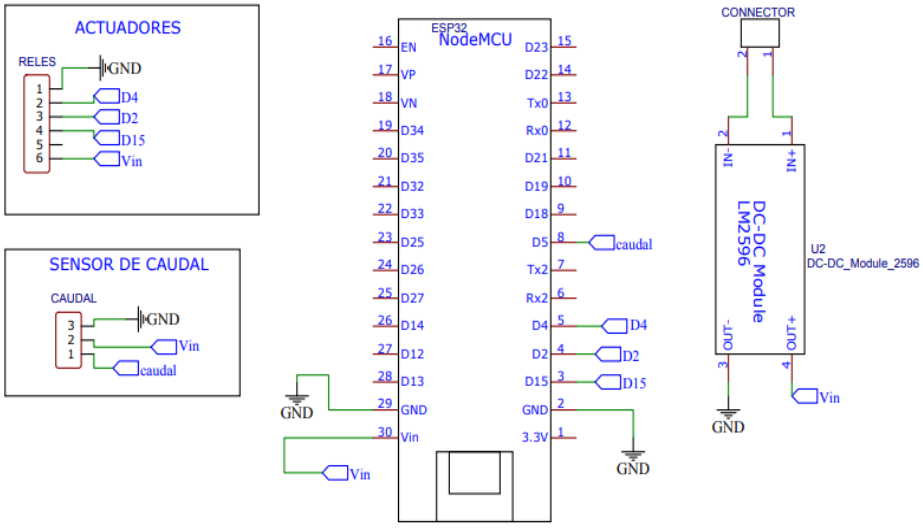


Elaborado por: Estefania Acosta & Carlos Cuaical.

2.7.4. Circuito eléctrico para la irrigación de un huerto casero

Para los diseños de los circuitos del proyecto se ha utilizado el programa EasyEDA, se ha realizado 3 circuitos, en el primero como se muestra en la Figura 20, se va a controlar las electroválvulas mediante un módulo relé, se obtendrá el valor del consumo de agua en litros por medio del sensor YF-S201, además se implementa un regulador de voltaje LM2596, ya que este admite voltajes de entrada de 4.5V a 40V y una corriente máxima de 3A, por lo que es posible alimentar el circuito con una fuente de 12V y ser regulada a 5V para la alimentación del circuito.

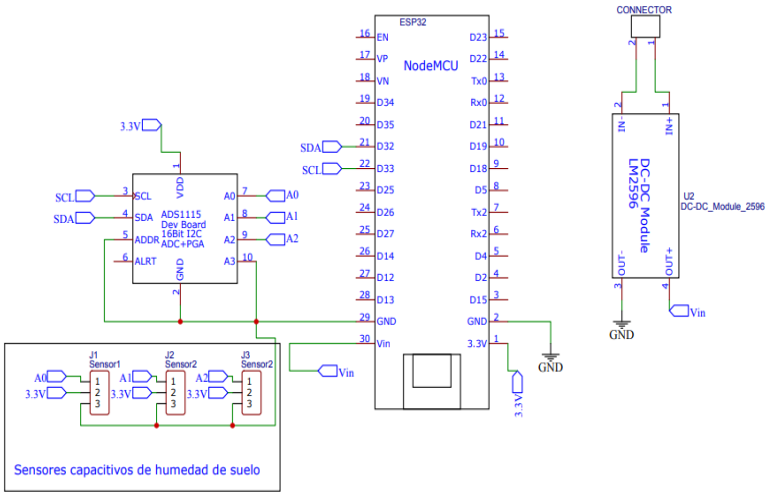
Figura 20: Circuito Eléctrico de Actuadores y el Sensor de Caudal.



Elaborado por: Estefania Acosta & Carlos Cuaical.

En el segundo circuito como se muestra en la Figura 21, se utiliza un conversor de voltaje como se mencionó anteriormente y además se implementa un módulo ADS1115, el cual permite obtener los valores análogos de los sensores de humedad de suelo capacitivos con una buena precisión, su ADC funciona con 16 bits programables, 4 pines para lecturas análogas y se comunica con el microcontrolador mediante I2C, este módulo es importante, porque el Esp32 comparte su ADC con el wifi por lo que no es recomendable usar el conversor ADC y el driver del wifi al mismo tiempo ya que existe interferencia entre ellos.

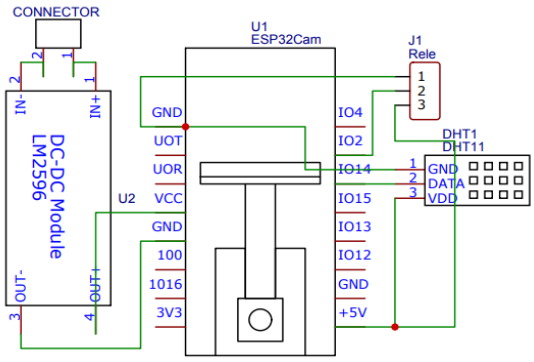
Figura 21: Circuito Eléctrico de los sensores capacitivos.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Para el último circuito como se puede observar en la Figura 22, se utiliza un conversor LM2596 para la alimentar el circuito con 5V, se recolectan valores de humedad y temperatura del ambiente con el sensor Dht11 y se implementa un módulo relé para la activación de un foco dentro del huerto casero.

Figura 22: Circuito Eléctrico del sensor dht11 y una luminaria.



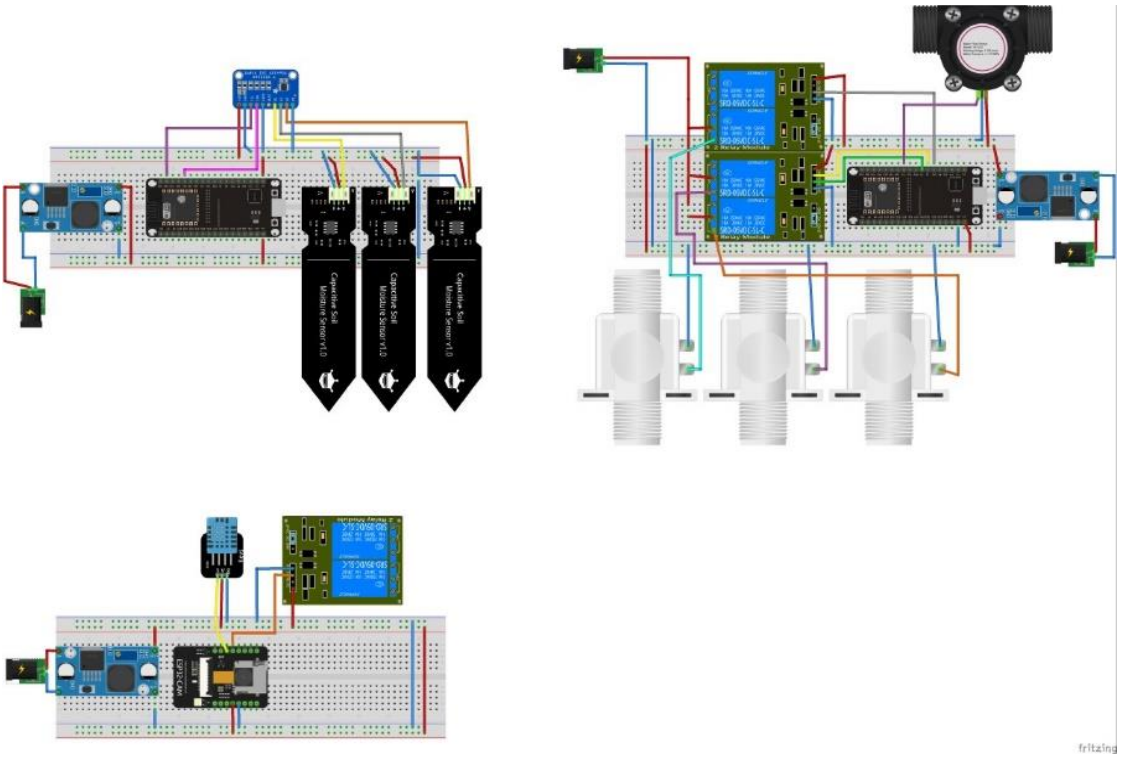
Elaborado por: Estefania Acosta & Carlos Cuaical.

2.7.5. Diagrama Esquemático del sistema automático de irrigación

En la Figura 23, se representa el diagrama esquemático, el cual ha sido diseñado con el programa Fritzing, en el cual se puede visualizar que se ha hecho uso de algunos materiales como: el sensor Dht11, sensor capacitivo de humedad de suelo, relés,

electroválvulas, sensor de caudal, Esp32 y regulador LM256, todos estos con sus respectivas conexiones para la automatización del huerto casero.

Figura 23: Diagrama Esquemático del Sistema automático de riego.

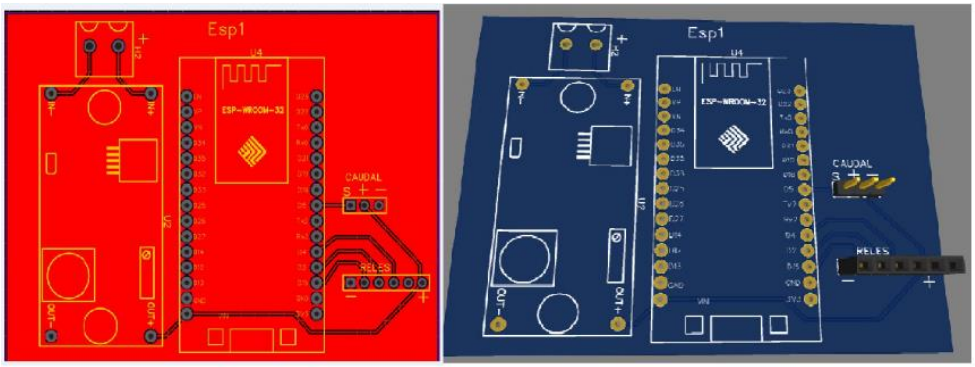


Elaborado por: Estefania Acosta & Carlos Cuaical.

2.7.6. PCB del Sistema automático de irrigación

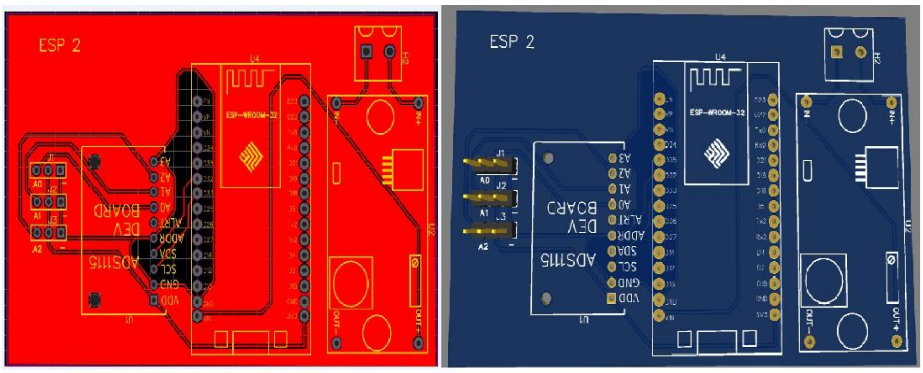
Para el diseño de las PCB´s de este proyecto se ha utilizado el programa EasyEDA como se muestra en la siguiente Figura 24, se realizó 3 circuitos para este proyecto, la primera placa es para la conexión de los sensores capacitivos, que consta de un módulo Esp32, con espadines para instalar el sensor de caudal y el relé, además del regulador de voltaje, los cuales están conectados por medio de vías de cobre. Las placas del módulo 2 que contiene las electroválvulas se las puede observar en la Figura 25 y módulo 3 el cual activa una luminaria y tiene el sensor Dht11 en la Figura 26.

Figura 24: Placa PCB del módulo 1 con los sensores capacitivos



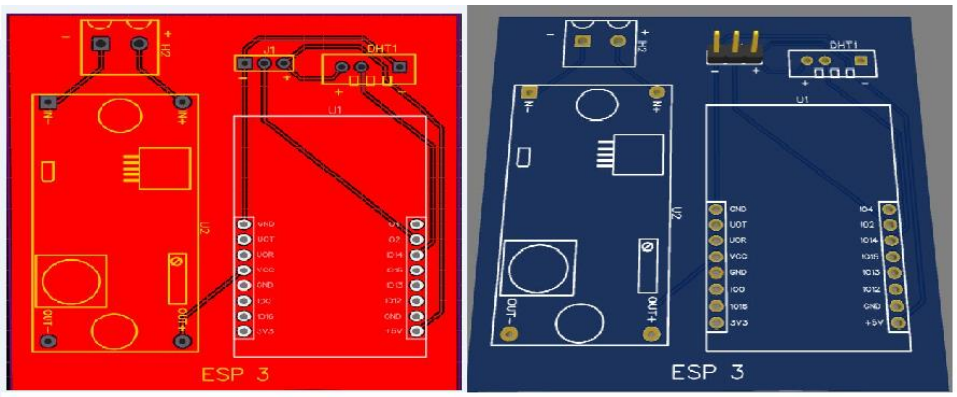
Elaborado por: Estefania Acosta & Carlos Cuaical

Figura 25: Placa PCB del módulo 2 con las electroválvulas.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Figura 26: Placa PCB del módulo 3.



Elaborado por: Estefania Acosta & Carlos Cuaical.

CAPÍTULO 3

3. IMPLEMENTACIÓN DEL PROTOTIPO

3.1.Instalaciones en la Raspberry pi

3.1.1. Instalación de Openhabian en Raspberry Pi

En el sitio web oficial de Openhab se encuentra disponible muchas distribuciones de instalación para sistemas operativos, se debe seleccionar uno dependiendo cual sea la mejor opción para la automatización. Existe una imagen estable recomendada para los usuarios que usen Raspberry Pi, para su instalación se debe realizar lo siguiente:

- Descargar una imagen ISO estable de Openhabian del sitio web oficial de Openhab.
- Cargar la imagen descargada previamente en una memoria SD de 32GB de memoria usando el programa Rufus.
- Cuando se termine de copiar la imagen en la tarjeta de memoria SD, insertarla en la Raspberry Pi.
- Conectar la Raspberry Pi al servicio de internet mediante cable ethernet.
- Energizar la Rasberry Pi mediante una fuente de 5V que entregue más de 2 amperios.
- Esperar unos minutos hasta que se instale la imagen en la Raspberry pi.
- Asignar una dirección IP estática a la Raspberry pi, ingresando a la configuración del router doméstico, en este caso se ingresó mediante la dirección 192.168.1.1.
- El nombre del dispositivo aparecerá como Openhabian y a él se le debe colocar la IP estática que se desee, en este caso se ha utilizado la dirección IP 192.168.1.11.
- Ingresar la IP estática asignada con la terminación :8080 en un buscador de internet para visualizar la pantalla de inicio de Openhab como se muestra en el Anexo 1.
- Ingresar los parámetros de usuario y contraseña que se desea para crear una cuenta de administrador.
- Configurar los siguientes parámetros que aparecen como idioma, ubicación, región y zona horaria según los requerimientos que se necesite.
- Instalar MQTT Binding.

3.1.2. Instalación de Mosquitto MQTT en la Raspberry pi

La imagen ISO instalada cuenta con algunos componentes que serán útiles para la automatización, uno de ellos es Mosquitto que sirve para convertir la Raspberry pi en un Bróker, para que pueda interactuar con los otros dispositivos que serán los clientes. Para instalarlo se deben efectuar los siguientes pasos:

- Ingresar mediante SSH a la Raspberry pi utilizando PuTTY como se muestra en el Anexo 2.
- Ingresar el usuario y la contraseña las cuales por defecto son Openhabian.
- Entrar a la configuración mediante el comando que se muestra en el Anexo 3.
- Seleccionar la opción 20 como se muestra a continuación en el Anexo 4.
- Elegir la opción 23 para instalar Mosquitto como se muestra en el Anexo 5.

3.1.3. Instalación de la base de datos InfluxDB

Para poder almacenar los datos de este proyecto se utilizará la base de datos InfluxDB, la cual es compatible con Openhab y se instala de la siguiente manera:

- Digitar el comando que se muestra en el Anexo 6.
- Para que InfluxDB funcione como un servicio se usa el comando del Anexo 7.
- Una vez realizado estos pasos se debe reiniciar el servicio con el comando que se muestra a continuación en el Anexo 8.
- Ingresar en el fichero influxdb.conf con el comando que se muestra en el Anexo 9.
- Activar la autenticación http como se muestra a continuación en el Anexo 10.
- Crear una base de datos y los usuarios con su contraseña respectiva como se muestra en el Anexo 11.
- Asignar los permisos para la base de datos con el comando que se muestra en el Anexo 12.
- Conectar Openhabian con la base de datos ingresando al fichero influxdb.cfg como se muestra en el Anexo 13.
- Ingresar la url, usuario, contraseña y el nombre de la base de datos que se creó anteriormente como se muestra en el Anexo 14.

- Abrir Openhab en un buscador, dirigirse a configuración, add-ons: persistence services y seleccionar InfluxDB como se muestra en Anexo 15.

3.2.Vinculación del proyecto a la nube

Openhab pone a disposición de los usuarios un servicio de control remoto que es gratuito y seguro mediante el cual el usuario puede acceder a su servidor y a su interfaz gráfica desde cualquier parte del mundo. (Saxena et al. 2019)

Para activar este servicio es necesario realizar los siguientes pasos:

- Ingresar al sitio web https://myopenhab.org/logih.
- Colocar un mail y contraseña.
- Mediante SSH ingresar al fichero que se muestra a continuación en el Anexo 16.
- Copiar el código que se muestra en el fichero y pegarlo en sitio web, en la parte que dice secret.
- Luego ingresar al fichero con el comando que muestra en el Anexo 17.
- Copiar el código y pegarlo en el sitio web, en el lugar que dice Openhab UUID.

3.3.Programación de los módulos Esp32

3.3.1. Programación del primer módulo Esp32

Este proyecto utiliza tres módulos Esp32 los cuales cumplen diferentes funciones en este huerto casero, el primero de ellos es encargado de recibir los mensajes del bróker para la activación de las electroválvulas y enviar el valor de la cantidad diaria de agua que se ha entregado al huerto, el segundo módulo envía hacia el bróker el porcentaje de humedad del huerto, mientras que el tercer módulo es el encargado de activar la iluminación para el huerto y también envía los valores de humedad y temperatura del entorno.

Los tres módulos se han programado usando el IDE de Arduino, y se ha utilizado la librería PubSubclient la cual admite publicar y suscribirse a los mensajes del protocolo MQTT, esta librería es compatible con los módulos que se utilizan en el proyecto. (Lafuente 2019)

La librería Wifi.h es utilizada para poder conectar los Esp32 al router que está conectado al servicio de internet del domicilio y la librería ntpclient.h es usada para obtener la hora local y la fecha actual según la ubicación regional del equipo, la información la obtiene de un servidor usando el protocolo Network Time Protocol (NTP).

En el Anexo 18, en las primeras líneas de código de la 6 a la 12 se muestra el ajuste de los parámetros necesarios para obtener los días de la semana en español y también la hora de acuerdo con la zona horaria, en el caso de este proyecto se ha colocado el valor -1800, que corresponde a la zona horaria de Ecuador.

En las líneas 44 hasta la 53 del Anexo 18, se muestra la configuración de la dirección estática que se le ha asignado al primer Esp32, se debe ingresar la SSID y la contraseña del router para conectar el dispositivo por medio de Wifi y se ingresa la dirección IP estática del Bróker. Cada dispositivo que se desee conectar al bróker debe tener un único nombre, para este proyecto el primer Esp32 se le ha asignado el nombre de esp1cliente.

La función callback que sirve para que el dispositivo se subscriba a los temas que envía el bróker, dentro de esta función se debe escribir las acciones que se desea realizar,

cuando se obtiene un determinado mensaje que es enviado por el bróker. En este dispositivo se ha programado el accionamiento de las electroválvulas encargadas de la irrigación del huerto.

En las líneas 156 hasta la 181 del Anexo 18 se muestra, el código que se utiliza para reconectar el dispositivo al Bróker en caso de que se desvincule de él, cuando este acontecimiento sucede por medio de la publicación Last Will, se puede determinar si el dispositivo se encuentra Online u Offline. En este apartado también se debe programar los mensajes para la suscripción de dispositivos.

En la función loop se programa la lectura del caudal de flujo y las condiciones en las que se activarán las electroválvulas, la programación completa de esta función se la puede ver en el Anexo 18.

3.3.1.1. Calibración de sensores de humedad capacitivos por el método de gravimetría

El método de calibración por gravimetría es el recomendado para los sensores de humedad capacitivos, este método consiste en medir el volumen de agua (V_A) que contiene una muestra de un determinado volumen de suelo (V_s), la humedad volumétrica se simboliza con el símbolo griego θ y se lo define como se muestra en la Ecuación 1. (Radi et al. 2018)

La masa del agua puesto que no es conocida se la puede obtener por medio de una resta entre la masa de la muestra de tierra húmeda (m_{sh}) menos la masa de la muestra seca (m_s).

$$\theta = \frac{V_A}{V_s} = \frac{\frac{m_A}{\rho_A}}{\frac{m_S}{\rho_S}} = \left(\frac{m_{sh}-m_s}{m_s}\right)\frac{\rho_s}{\rho_A} \qquad \text{EC. 1}$$

La densidad del agua se considera como 0.997 $\left[\frac{k_g}{L}\right]$ y la densidad del suelo se la obtendrá dividiendo la masa del suelo seco y el volumen de la muestra.

Pasos para la calibración:

- Con una balanza medir la masa de un envase que pueda contener 200 ml de una muestra de suelo.
- Llenar el envase con una muestra de suelo seco hasta alcanzar los 200 ml, la muestra debe estar completamente seca para ello se debe dejar secar la tierra algunos días.
- Medir en la balanza la masa de la muestra en el envase.
- Verter diez mililitros de agua en la muestra y mezclar muy bien.
- Medir la masa de la muestra húmeda.
- Insertar el sensor de humedad en la muestra y medir cuanto voltaje entrega.
- Se debe repetir los pasos 4, 5 y 6 hasta que la muestra de suelo se rebose de agua.
- Tomar la muestra húmeda y dejarla secar, este paso tomara varios días dependiendo de las condiciones climáticas en las que se encuentre.
- Cuando la muestra este completamente seca hay que medir su masa con la balanza y con ese valor se debe determinar la densidad del suelo.

En el Anexo 19 se muestra los parametros iniciales como la masa del recipiente, densidad del agua, volumen , densidad del suelo y masa del suelo seco, para poder realizar la calibracion del sensor capacitivo de humedad de suelo.

En el Anexo 20 se visualiza los valores obtenidos por la calibración, para cada muestra se ha colocado 10ml de agua en la tierra seca, se fueron tomando los valores y realizando los respectivos cálculos matemáticos.

En el Anexo 21 se muestra la gráfica con la respectiva ecuación lineal de tendencia.

Se puede apreciar en la Ecuación 2, la ecuación de transferencia que se debe utilizar para el sensor 1.

$$\theta = \frac{0.9308}{V} - 0.3762 \qquad \text{EC. 2}$$

De igual manera se realizó el procedimiento para sensor 2 con el cual se obtuvo la gráfica del Anexo 22 y la ecuación de transferencia en la Ecuación 3.

$$\theta = \frac{0.7543}{V} - 0.3593 \qquad \text{EC. 3}$$

Para el sensor 3 se obtuvo la gráfica del Anexo 23 y la ecuación de transferencia que se muestra en la Ecuación 4.

$$\theta = \frac{0.6193}{V} - 0.2444 \qquad \text{EC. 4}$$

Una vez realizada la calibración, se procedió a comprobar las funciones de transferencia obtenidas llenando una muestra de suelo seco de 200ml y se colocó 40ml de agua para la verificación de la calibración de los sensores realizada. Los porcentajes de humedad de suelo que registraron los sensores capacitivos fueron similares como se puede apreciar el Anexo 24 con lo cual se puede decir que la calibración y las funciones de transferencia obtenidas fueron exitosas por medio del método de gravimetría.

3.3.2. Programación del segundo módulo Esp32

El segundo módulo es el encargado de recolectar los valores censados por los sensores de humedad capacitivos y por medio de una calibración por el método gravimétrico enviar los valores de humedad de suelo detectados, estos valores se envían en forma de porcentajes para poder comprender mejor la cantidad de humedad que existe en el suelo. (Hrisko Joshua 2020)

Para obtener los valores análogos de los sensores se utiliza el módulo Ads1115 con el cual se pueden obtener valores análogos más precisos y permite conectar hasta 4 sensores de humedad. Este módulo se utiliza con las librerías que se puede descargar desde el IDE de Arduino, en donde se debe especificar la dirección y el factor de escala con el cual se desea trabajar, en este caso se utilizara la dirección 0x48 ya que el pin ADDR del módulo Ads1115 está conectado a tierra y se usa el factor de escala 0.125f ya que la señal que envían los módulos de humedad de suelo capacitivos van desde 260 Hz cuando la tierra está muy húmeda hasta 520 Hz cuando la tierra está seca.(Radi et al. 2018)

En la función setup se ingresa la ganancia con la cual se desea que funcione el ADS1115 que para este dispositivo será la ganancia de uno.

La programación completa más detallada se encuentra en el Anexo 25, aquí también se puede ver que dentro de un bucle for se realiza la lectura de los valores análogos de los sensores con la finalidad de obtener la moda, para tener datos más estables, se puede apreciar la programación de los pines análogos conectados a los sensores, así como también el uso de la funciones de transferencia obtenidas, se puede observar el almacenamiento de los valores obtenidos de la moda de cada sensor, en una variable

float, estos valores son posterior mente transformados a String para ser enviados al bróker cada 2 segundos

3.3.3. Programación del tercer Esp32

El tercer Esp32 es el encargado de obtener la temperatura y humedad del entorno utilizando el sensor DHT11 y también debe encender una luminaria que está ubicada en el huerto. Para configurar este dispositivo es necesario importar la librería del sensor la cual se la puede descargar desde el IDE de Arduino. En este módulo también se aprovecha la cámara que viene incluida y se lo configura de tal forma que pueda ser utilizado con Telegram, para esto se ha creado un Bot y se incluye la librería universal que ofrece Telegram, el código completo de este módulo se muestra en el Anexo 26.(Junior et al. 2019)

Se ingresa las credenciales del router al cual se quiere conectar, así como también la dirección IP del bróker, se coloca un nombre que no tengan los otros dos dispositivos, en este caso se llamará esp3Cliente y se asigna el pin en el que estará conectado el sensor DHT11. En la función loop se configura él envió de los datos de humedad y temperatura en grados centígrados cada 5 segundos.

3.4.Configuración de un fichero Things en Openhab

Un Thing o cosa en español es posible incorporar al sistema de manera física o también puede ser virtual como un servicio web el fichero things podría contener características de configuración de acceso a algún servicio o algún tipo puntual de arreglo que cambia de alguna manera el funcionamiento del dispositivo.(openHAB 2019b)

Dentro de las cosas existen Channels o canales que son las distintas actividades concretas que realiza un Thing. Los canales están ligados a los Items o elementos los cuales son los que unen la capa virtual y la física. Las cosas reaccionan a eventos enviados por los elementos.(openHAB 2019b)

Un Bridge o puente es una especie de Thing que se añade al procedimiento para tener acceso a otros Things.(openHAB 2019b)

Para crear un fichero Things es necesario ingresar mediante SSH a la Raspberry como super usuario e ingresar el comando que se muestra en el Anexo 27 en donde se debe crear un fichero de cualquier nombre pero que tenga la extensión. things, en el caso de este proyecto se ha creado el fichero con nombre mymqtt.things.

Dentro de este fichero lo primero que se debe realizar es un Bridge para comunicar el Bróker con los clientes, este Bridge de ser configurado como se muestra en el Anexo 28, en donde se especifica el nombre del Bridge, la dirección IP con la que está configurada la Raspberry Pi, el puerto 1883 con el que funcionara el protocolo MQTT y el clientID.

A continuación, se crea las cosas a las que se debe conectar el bróker como se muestra en el Anexo 29, en donde topic sensores es la identidad que se le otorga al Thing, “Humedad del suelo” es el nombre de la cosa y “Huerto” es la ubicación de la cosa.

Para crear los canales es necesario poner el tipo de variable pueden ser de tipos como, por ejemplo: string, number o switch, seguido del nombre de identificación, un rotulo, le mensaje al cual debe suscribirse el canal con el comando stateTopic y el mensaje que se desea publicar con el comando commandTopic, en el Anexo 30 se puede visualizar la configuración de canales para la recepción de los mensajes del módulo esp2 mediante el protocolo de comunicación MQTT.

3.5.Creación del fichero Items

El fichero de elementos Items se crea ingresando el comando que se visualiza en el Anexo 31 mediante SSH, en dónde el fichero puede tener cualquier nombre, pero su extensión debe ser. ítems, en este ejemplo se ha creado el fichero Huerto.items.

Para crear un elemento dentro del fichero ítems se sebe especificar primero el tipo de item ya sea number, string, switch, entre otros, seguido del nombre del Item, el nombre del texto, el icono, el grupo y la configuración del canal al cual se desea enlazar como se muestra en el Anexo 32.

3.6.Creación del fichero Persistence

El fichero Persist sirve para almacenar los datos en la base de datos InfluxDB, para su creación es necesario establecer un nombre de fichero que tenga la extensión ". persist" como se muestra en el Anexo 33 en la cual se ha creado un fichero con el nombre "influxdb.persist."

Dentro del fichero creado es necesario colocar la frecuencia con la cual se desea almacenar los datos en InfluxDB para lo cual se debe colocar la sintaxis como se

muestra en el Anexo 34 eso permitirá seleccionar el tiempo de almacenamiento en función de cada minuto, hora, día o cada que se genere un cambio de estado.

El siguiente apartado que se debe configurar los elementos que se desea guardar en la base de datos, lo cual se hace como se muestra en el Anexo 35 en donde primero escribimos el Ítem que se desea almacenar seguido de la estrategia, es decir con qué frecuencia se desea almacenar el dato del elemento seleccionado.

3.7.Creación del fichero Sitemaps

Openhab ofrece varias interfaces gráficas las cuales pueden ser configuradas por el usuario de acuerdo con su gusto y conveniencia. El fichero Sitemaps sirve para crear una interfaz gráfica básica llamada BasicUI, para su creación se debe ingresar el comando que se muestra en el Anexo 36. El fichero puede tener cualquier nombre, pero su extensión debe ser .sitemap, en este caso se le ha asignado el nombre de panelbasicol.sitemap. En el Anexo 37, se muestra la programación del panel básico que se ha realizado en este proyecto. En el Anexo 38 se muestra la visualización que tiene el usuario de la BasicUI programada.

3.8.Creación del fichero Rules

El fichero Rules o reglas en español, sirve para programar la automatización que se desee realizar en el proyecto, su programación se basa en Xbase. Las reglas deben ser colocadas en un fichero con extensión .rules, en Anexo 39 se muestra el fichero creado para este proyecto.

Las reglas deben tener una sintaxis exclusiva para su funcionamiento, a continuación, en el Anexo 40 se muestra la regla generada para la automatización de proyecto. En el

lugar que dice rule se debe poner el nombre de la regla entre comillas, en el parámetro When se programa cuando se necesita que la regla se ejecute en este caso se ha realizado cada vez que el porcentaje de los sensores cambie, en Then se debe situar la condición que se cumplirá, en el caso de este proyecto se han enviado valores numéricos para que el segundo módulo Esp32, el cual controla las electroválvulas y realiza una selección de casos mediante la condición switch case.(openHAB 2019a)

3.9.Creación de una interfaz gráfica mediante Widgets

Al instalar Openhab se crea una página llamada Overview, la cual no se puede borrar, pero si se puede configurar al gusto del usuario. Para configurar esta página se debe dirigir a la pestaña herramientas de desarrollo y después seleccionar la opción Widgets, allí se debe dar clic en el símbolo de "+" y se desplegara un ejemplo de Widget en donde se puede configurar el Widget mediante código YAML, para lo cual se puede guiar del demo de demostración que ofrece Openhab. En este proyecto se ha tomado personalizado algunos Widgets para poder crear una interfaz cómoda para el usuario.

En el Anexo 41 se muestra la configuración en código YAML que se ha realizado para mostrar el estado de un dispositivo Esp32. El mismo código YAML se puede utilizar para mostrar el estado de los otros dispositivos únicamente cambiando el Ítem y los respectivos nombres. (openHAB 2021)

Para la activación de una luminaria se ha configurado el código como se muestra en el Anexo 42, donde se puede observar que se ha configurado un Widget para que funcione como botón y envíe los comandos ON u OFF.

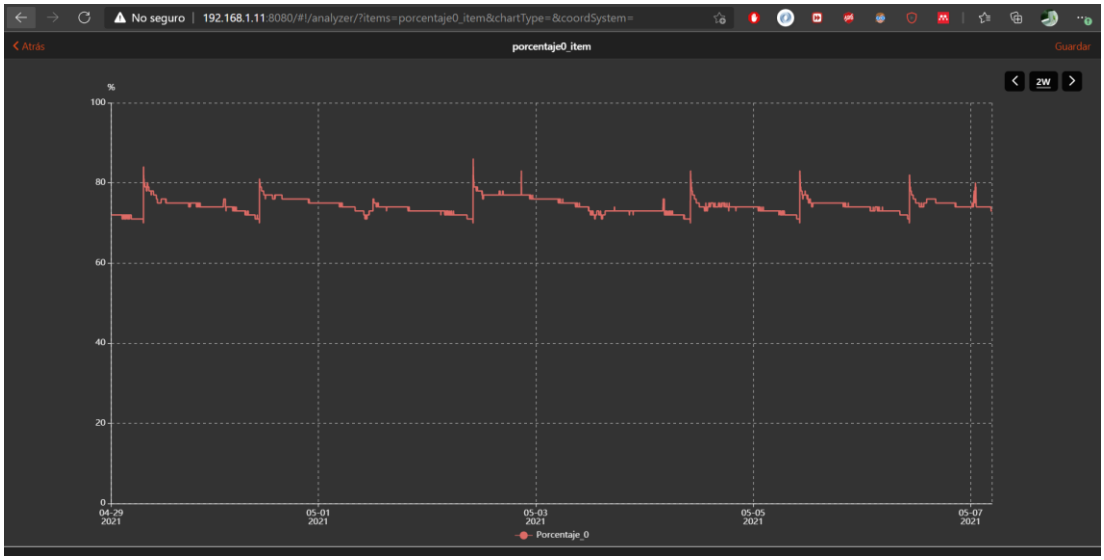
En la Figura 27 se muestra programado el Widget de tal forma que se pueda observar el porcentaje de humedad de suelo y al dar clic en el Widget ingrese a la gráfica con

los datos que se ha recopilado en la base de datos InfluxDB como se observa en la Figura 28.

Figura 27: Configuración de Widget con Gráfica.

Elaborado por: Estefania Acosta & Carlos Cuaical.

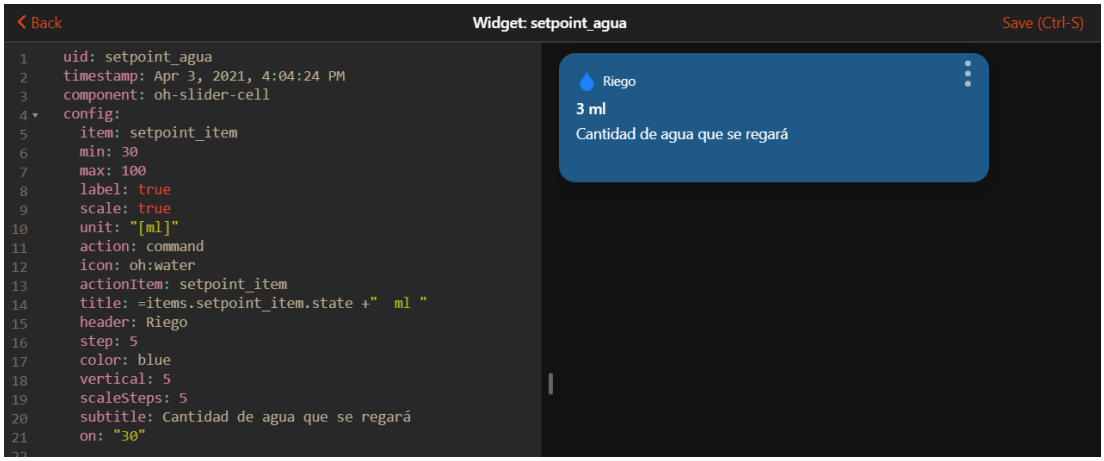
Figura 28: Gráfica del porcentaje del sensor 1.



Elaborado por: Estefania Acosta & Carlos Cuaical.

En el siguiente Widget de la Figura 29 se ha programado un botón para enviar el dato del setpoint para seleccionar el valor de la cantidad de agua que se debe regar.

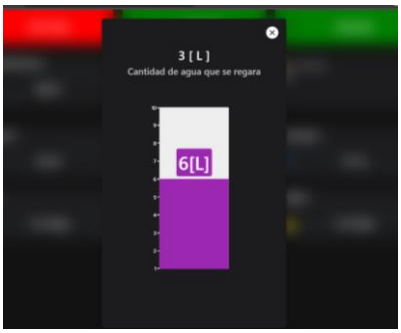
Figura 29: Configuración del Windget para seleccionar la cantidad de agua que se regara.



Programación del Windget en código YAML en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical.

En este Widget al dar clic en los tres puntos se despliega un potenciómetro para que el usuario pueda seleccionar el valor que desee como se muestra en la Figura 30

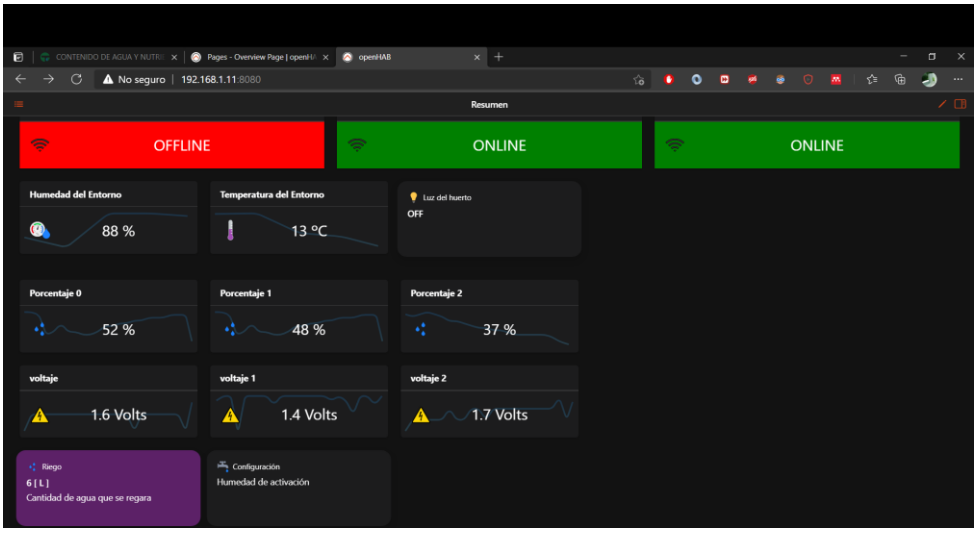
Figura 30: Selección de cantidad de agua.



Selección de cantidad de agua en la interfaz gráfica de Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical.

La interfaz gráfica final se puede apreciar en la Figura 31.

Figura 31: Interfaz Gráfica en Openhab.



Visualización de la página principal en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical.

3.10. Configuración de alertas

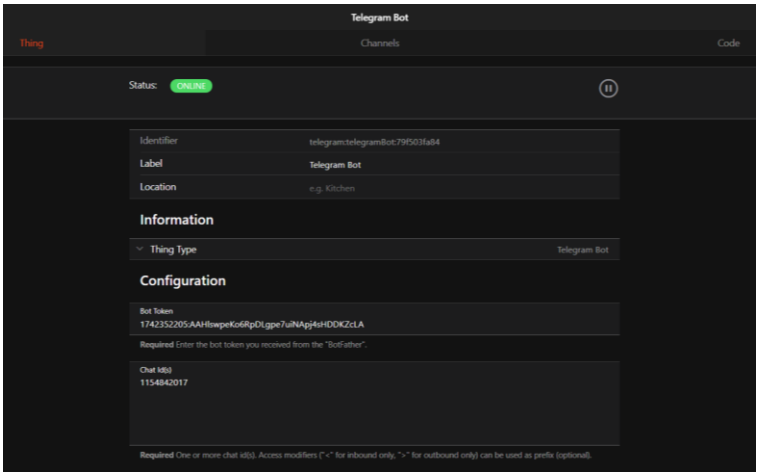
Es necesario que el usuario tenga conocimiento de los estados de los dispositivos para saber si el sistema está funcionando correctamente, pueden existir diferentes factores por los cuales los dispositivos pueden desconectarse ya sea por falta de energía o por algún daño. (Kumar and Silva 2020) En Openhab es posible enviar mensajes a un ChatBot creado en Telegram para lo cual es necesario ingresar a la configuración e instalar Telegram Binding, luego se debe ingresar a Telegram, en el buscador de Telegram se debe buscar BotFather, como se muestra en la Figura 32. Se muestran diferentes opciones para crear un Bot, en este caso se selecciona "/newbot" para la creación de un nuevo Bot, se selecciona el nombre del Bot y después se recibe un token de acceso. (Salvi, Geetha, and Sowmya Kamath 2019)

Figura 32: Creación de ChatBot en Telegram con BotFather.

Opciones que ofrece BotFather, Elaborado por: Estefania Acosta & Carlos Cuaical.

En la configuración de Openhab se ingresa a Things se busca Telegram Bot y dentro de esta opción se digita el token que fue proporcionado anteriormente por BotFather al igual que el Chat Id del Bot creado como se muestra en la Figura 33.

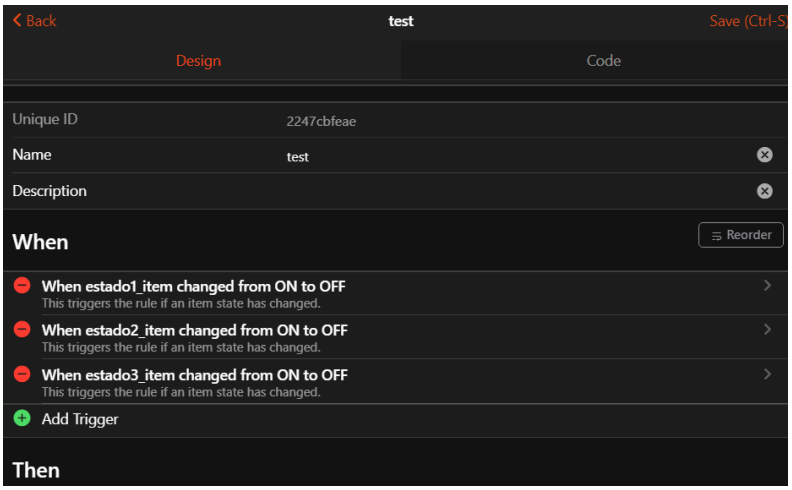
Figura 33: Configuración de Telegram Bot en Openhab.



Configuración del token y del Chat id en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical.

Después de esto hay que dirigirse a rules y aquí se debe seleccionar los Items que son encargados de obtener la información de estado de conexión de los dispositivos, se debe seleccionar el estado previo y estado cuando se deba activar la alarma, en este caso requiere que se active la alarma cuando el estado cambie de ON a OFF, como se muestra en la Figura 34.

Figura 34: Condiciones para la activación de la alerta.



Configuración de la regla en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical.

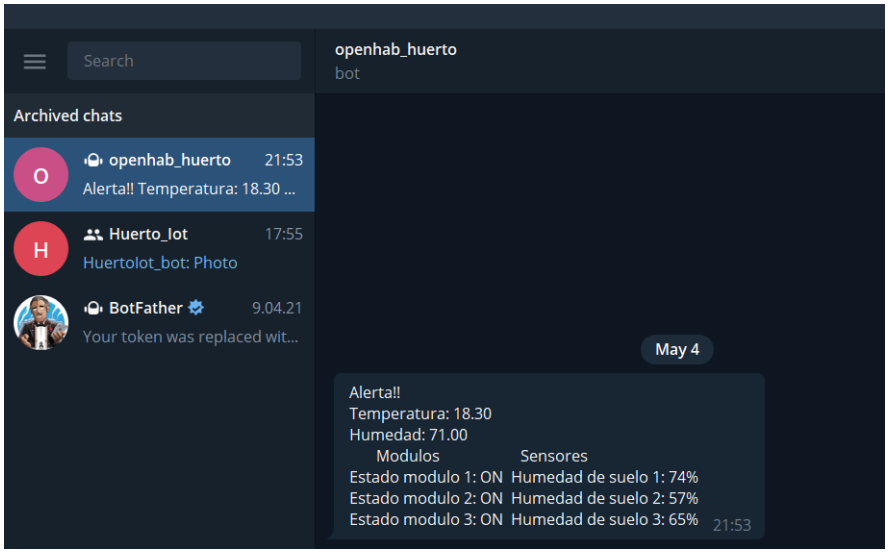
Después de esto se debe programar un script para que se envié la alerta al ChatBot creado en Telegram, este script se puede poner los items que se quiera mostrar en el mensaje como se muestra en la Figura 35.

Figura 35: Script se ejecutará cuando se active la regla.

Script en Openhab que se ejecutara cuando funcione la regla, Elaborado por: Estefania Acosta & Carlos Cuaical.

En la Figura 36 se muestra el mensaje que se recibe cuando se activa la alerta

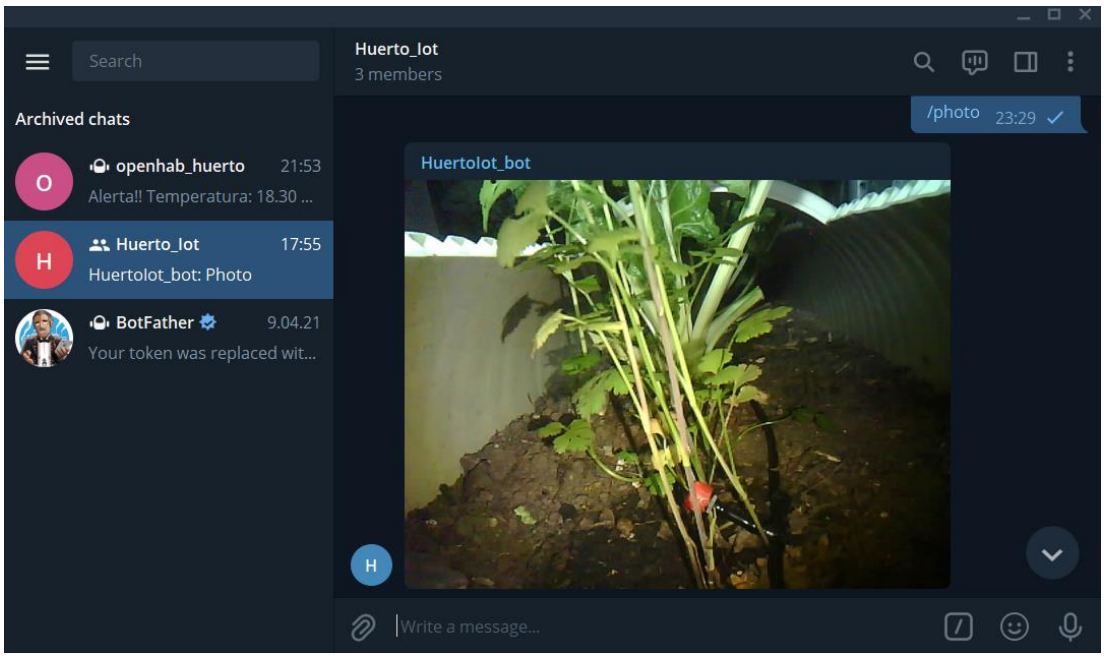
Figura 36: Ejemplo de Alerta recibida al ChatBot.



Mensaje recibido al Bot Creado en Telegram, Elaborado por: Estefania Acosta & Carlos Cuaical.

y también se puede ver en la Figura 37 que cuando se escribe el comando "/photo" en el Bot creado en Telegram, se recibe una fotografía en tiempo real. (Muslih et al. 2019)

Figura 37: Obtención de Fotografía por medio del ChatBot.



Petición de Foto desde el ChatBot de Telegram, Elaborado por: Estefania Acosta & Carlos Cuaical.

CAPÍTULO 4

4. ANÁLISIS DEL SISTEMA AUTOMÁTICO DE IRRIGACIÓN

4.1.Análisis de Costos

4.1.1. Estudio de costo beneficio del sistema de riego IoT

Existen varias maneras de evaluar la factibilidad de un negocio o emprendimiento, para esta investigación se ha seleccionado, el análisis de costo beneficio, ya que es el más utilizado en el ámbito de proyectos, para poder determinar si la inversión que se planea llevar a cabo es rentable o no, por estos motivos se debe calcular el costo por unidad fabricada y el beneficio por parte de la venta de los productos, para ello se ocupa la siguiente fórmula Ecuación 5. (Perez 2015)

$$C/B = \frac{VAI}{VAC} \qquad \text{EC. 5}$$

Donde:

VAI: Valor actual de los ingresos totales netos o beneficios netos

VAC: Valor actual de los costos de inversión o costos totales

Decisiones de la razón costo beneficio:

- C/B > 1 El proyecto es rentable
- C/B < 1 El proyecto no es rentable
- C/B = 1 El proyecto no es rentable

Después de realizar los cálculos correspondientes, se compara con cada una de estas decisiones y por medio de estas se podrá determinar si el proyecto es viable o no. (Mahmud and Nafi 2020)

4.1.2. Costo de componentes

En la Figura 38, se presentan los materiales utilizados en este prototipo con sus respectivos precios unitarios para la implementación del sistema automático de riego.

Figura 38: Precios de componentes.

Unidad	Componentes	Precio unitario	Precio total
3	Sensor Capacitivo	4	12
2	Esp32	13	26
1	Esp32 CAM	15	15
3	Módulos Relé	3	9
1	Sensor ADS1115	5	5
3	Regulado de voltaje	2	6
1	Sensor de caudal	8,75	8,75
3	Electroválvulas	9,95	29,85
1	Sensor Dth11	3	3
3	Manguera	7	21
1	Raspberry pi	140	140
3	Placas Pcb	1,5	4,5
		TOTAL	268,1

Se ha realizado la suma total de los precios de cada elemento, Elaborado por: Estefania Acosta & Carlos Cuaical.

En donde se puede ver que la suma total calculada de los dispositivos que se utilizarán en este proyecto es de $268,1 dólares, a continuación, en la Figura 39 se detalla cuanto se cobrará por mano de obra por cada persona que forma parte de esta implementación.

Figura 39: Precio de mano de obra.

Mano de obra			
Mano de obra directa	**Tiempo empleado**	**Precio por hora**	**Total**
Trabajador 1	64	1,66	106,24
Trabajador 2	64	1,66	106,24
		TOTAL	212,48

Descripción del precio de mano de hora por cada integrante, Elaborado por: Estefania Acosta & Carlos Cuaical.

Como se pudo observar anteriormente el costo de mano de obra por cada trabajador es de $ 1,66 dólares por hora, ya que se debe realizar la instalación del sistema automático de riego y la programación, por lo que en todo eso se tomó un tiempo de 64 horas, realizando los respectivos cálculos matemáticos se obtuvo el valor de $212,48 dólares, de la mano de obra.

4.1.3. Cálculo del costo

Como se pudo mirar anteriormente en la Figura 33, se ha realizado una suma total de los costos de cada elemento, pero a esta estimación se le debe aumentar el valor de la mano de obra, como se representa en la Ecuación 6, a este se calcula el 30% para la ganancia, como se muestra en las siguiente Ecuación 7, a este resultado se lo conoce como la rentabilidad.

$$268{,}10 + 212{,}48 = 480{,}58 \quad \text{EC. 6}$$

$$480{,}58 * 30\% = \$\ 144{,}17\ Rentabilidad \quad \text{EC. 7}$$

$$480{,}58 + 144{,}17 = \$\ 624{,}75\ por\ unidad \quad \text{EC. 8}$$

Como se mostró anteriormente en la Ecuación 8 fue el precio final para la venta del producto.

4.1.4. Cálculo del beneficio

Después de obtener el valor de todos los componentes más la mano de obra, se obtuvo $ 624,75 dólares por unidad, que es el costo del producto, como se mostró en la Ecuación anterior. Por lo que se realiza una proyección en años, es decir que en 1 año se espera vender 250 unidades, a continuación, en la Ecuación 9, se visualiza el precio total de ventas durante el año, a este resultado se lo conoce como beneficio.

$$144{,}17x250 = \$\ 36042{,}5\ dólares \qquad \text{EC. 9}$$

4.1.5. Cálculo de relación costo beneficio

El valor de venta por cada producto es de $ 624,75 dólares, se espera vender 250 unidades por año, siendo el costo inicial del prototipo $ 480,58 dólares, calcular la relación de costo beneficio por medio del VAI y el VAC como se muestra en las siguientes Ecuaciones 10, 11 y 12.(Pérez 2015)

$$C/B = \frac{VAI}{VAC} \qquad \text{EC. 10}$$

$$C/B = \frac{624{,}75*250}{480{,}58*250} \qquad \text{EC. 11}$$

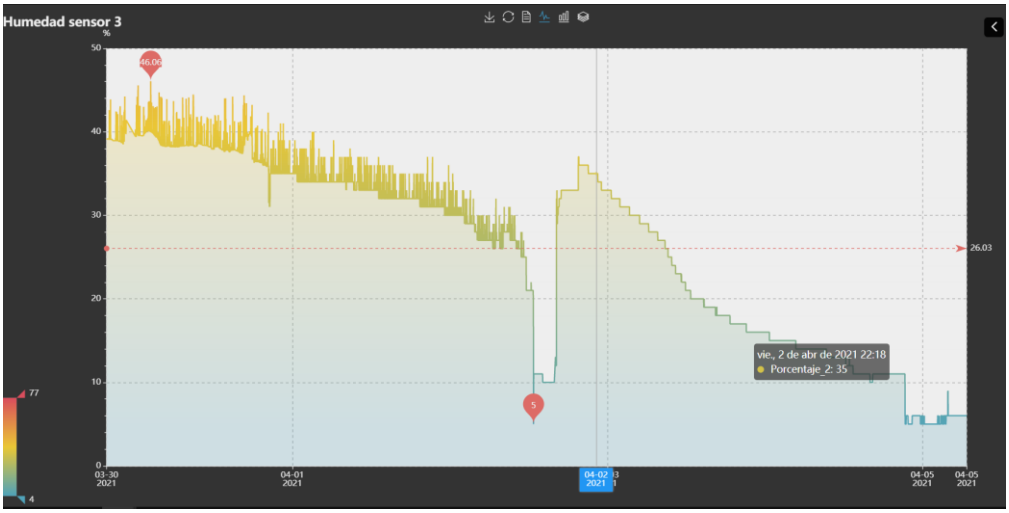
$$C/B = 1{,}29 \qquad \text{EC. 12}$$

Después de haber realizado todos los cálculos correspondientes, el valor de la relación es de 1,29 > 1, es decir el proyecto realizado es rentable.

4.2.Análisis de Resultados

Se recopilo los valores de humedad de suelo durante 5 días sin la implementación del sistema automático de riego como se muestra en la Figura 40, en donde se puede observar que el suelo los primeros días tenía una humedad aproximada del 37.63%, esta humedad va disminuyendo con hasta llegar a una humedad del 5% que es un valor muy bajo y es perjudicial para el cultivo, cuando llego a este valor se rego irrigo el suelo de forma manual y al cabo de dos días el suelo llego a tener otra vez una humedad del 4%.

Figura 40: Porcentaje de humedad de suelo del sensor 3 sin el funcionamiento del sistema de irrigación automático.



Elaborado por: Estefania Acosta & Carlos Cuaical.

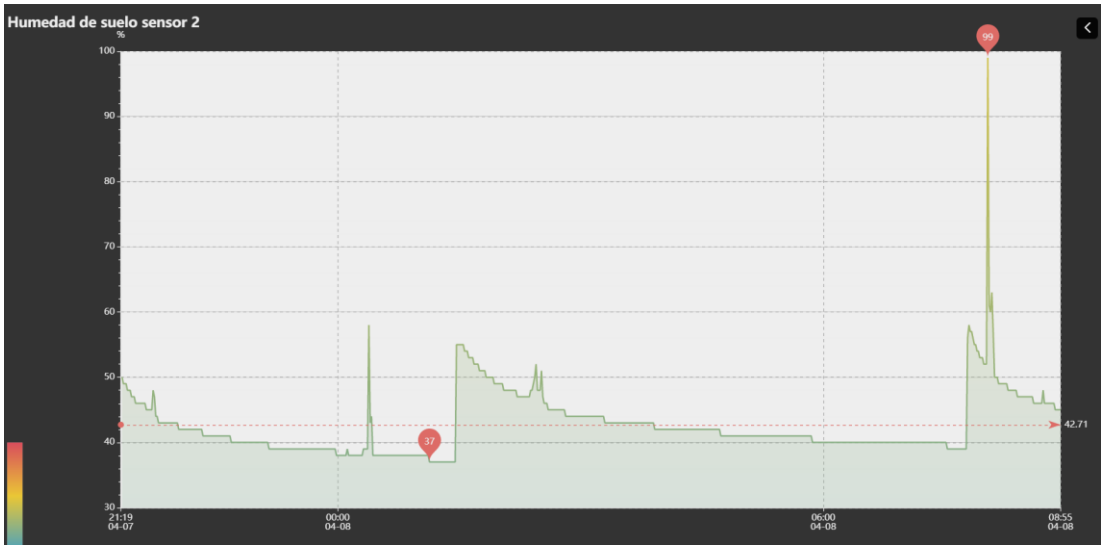
Después de que se implementó el sistema de riego automático se estableció en la interfaz gráfica que se riegue 60ml de agua con un valor de humedad de suelo umbral de 32% y luego se cambió el valor umbral de humedad de suelo a un 39% y un riego de 50ml como se muestra en la Figura 41 en la cual se puede apreciar que el huerto permanece bajo los parámetros programados y el promedio de humedad de suelo durante la muestra de 4 días es de 37.46, lo cual es favorable para el cultivo, ya que permanece en los rangos de humedad establecidos por el usuario.

Figura 41: Lecturas del sensor de humedad de suelo capacitivo con el sistema de irrigación automático.

Elaborado por: Estefania Acosta & Carlos Cuaical.

En ocasiones hay picos de voltaje que se generan al momento del riego, esto se debe a que el sensor necesita tiempo para estabilizarse y que el agua filtre por el suelo este fenómeno se puede ver en la Figura 42, la cual muestra el porcentaje de humedad de suelo del sensor 2.

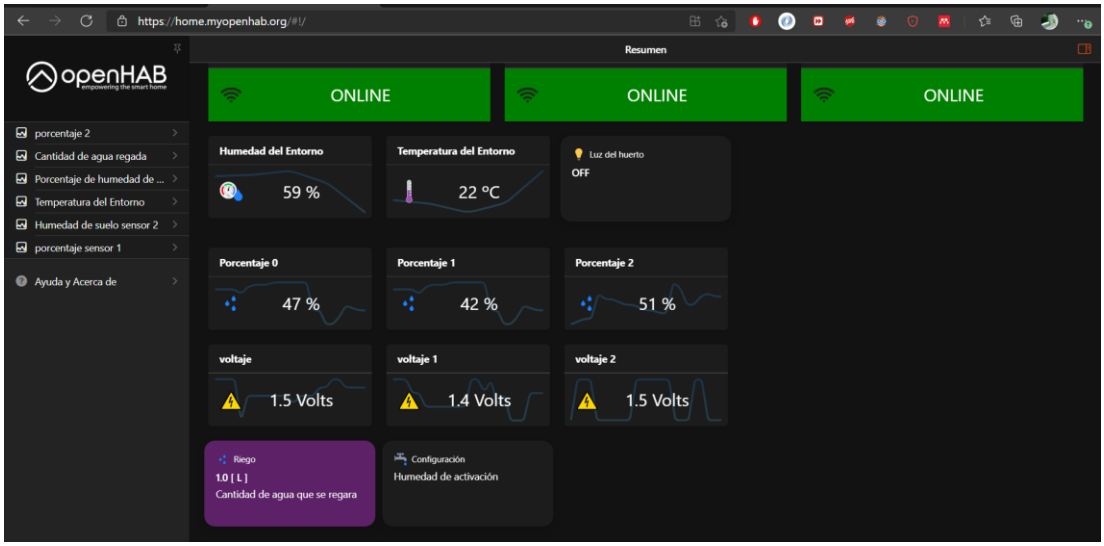
Figura 42: Porcentaje de humedad de suelo del sensor 2.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Mediante el portal web myopenhab.org fue posible el acceso al panel de control creado y se pudo apreciar los datos medio ambientales, los porcentajes de humedad de suelo y los almacenados en InfluxDB como se muestra en la Figura 43.

Figura 43: Ingreso a la interfaz gráfica mediante acceso remoto.



Elaborado por: Estefania Acosta & Carlos Cuaical.

CONCLUSIONES

El sistema automático de riego que se presenta en este documento utiliza sistemas embebidos y tiene una arquitectura IoT, realiza la irrigación automática tomando en consideración la adquisición de los valores de los sensores de humedad de suelo instalados en el huerto y los valores umbrales establecidos por el usuario

Mediante la investigación de trabajos previos sobre sistemas de irrigación automáticos fue posible determinar los componentes necesarios para la implementación de un sistema de irrigación para huertos caseros basado en IoT utilizando sistemas embebidos como una Raspberry Pi, módulos Esp32 y Openhab cuyo Software y Hardware son de código abierto lo que permite que el sistema sea de bajo costo.

El diseño del sistema de irrigación automático presentado en este proyecto permite monitorear en tiempo real la humedad de suelo del huerto casero, el usuario puede ingresar el porcentaje de humedad mínimo que desea que alcance el suelo, así como también la cantidad de agua que se requiere regar, todos los valores son almacenados en una base de datos privada. Este prototipo permite supervisar los datos de humedad y temperatura del medio ambiente, así como el encendido y apagado de una luminaria instalada en el huerto, todos estos valores pueden ser visualizados a través de una interfaz gráfica desarrollada en Openhab.

Mediante la creación de un ChatBot fue posible realizar alertas que indiquen cuando los dispositivos se desconectan del sistema, evitando de esta manera posibles problemas que se puedan presentar en la irrigación por falta de energía o algún problema en los dispositivos. El ChatBot también permite al usuario obtener una fotografía en tiempo real del huerto casero cuando la solicite, la calidad de imagen no es de alta definición, pero es suficientemente buena como para observar que la plantación se encuentra en buenas condiciones.

La implementación del prototipo se realizó de manera satisfactoria, la humedad del huerto se mantuvo en los valores umbrales establecidos por el usuario y su monitoreo mediante la interfaz gráfica fue útil. Mediante el acceso a la nube privada se pudo visualizar las variables de humedad de suelo, temperatura y humedad del medio ambiente almacenados en la base de datos y mediante el ChatBot creado en Telegram se logró obtener fotografías del huerto y alertas de desconexión de dispositivos.

Para el análisis de costos se tomó en cuenta la relación costo beneficio ya que es la más utilizada en el ámbito de proyectos, se realizaron los cálculos correspondientes y se determinó que por medio de la decisión costo beneficio el resultado obtenido de 1.29 define el prototipo implementado como económicamente factible.

RECOMENDACIONES

Openhab también incorpora la posibilidad de controlar la interfaz gráfica mediante voz, es recomendable investigar más sobre este tema para poder logar que usuarios con problemas visuales puedan hacer uso de la aplicación.

El Esp32 es un dispositivo de bajo consumo, la hoja de datos presenta diferentes modos de consumo que ayudarían al sistema a consumir aún menos energía. Se recomienda investigar al respecto para considerar implementar el sistema en escenarios donde exista problemas energéticos y se requiera alimentar el sistema con energías renovables como por ejemplo paneles solares o baterías.

Existen algunas mejoras que se pueden implementar a futuro como por ejemplo la instalación y programación de un sistema de fumigación o un sistema de detección de plagas mediante visión artificial.

El sistema de irrigación automático propuesto presenta una gran escalabilidad y podría ser usado para plantaciones grandes siempre que cuenten con cobertura de internet.

BIBLIOGRAFÍA

Assaf, Rasha, and Isam Ishaq. 2020. “Improving Irrigation by Using a Cloud Based IoT System.” *Proceedings - 2020 International Conference on Promising Electronic Technologies, ICPET 2020* 28–31.

Avşar, Ercan, Kurtuluş Buluş, Mehmet Ali Saridaş, and Burçak Kapur. 2018. “Development of a Cloud-Based Automatic Irrigation System: A Case Study on Strawberry Cultivation.” *2018 7th International Conference on Modern Circuits and Systems Technologies, MOCAST 2018* 1–4.

Choudhury, Amin, Soumen Moulik, and Sanjoy Choudhury. 2020. “Cloud-Based Real-Time and Remote Human Activity Recognition System Using Wearable Sensors.” 2020–21.

Fujii, Norihiro, and Nobuhiko Koike. 2017. “IoT Remote Group Experiments in the Cyber Laboratory: A FPGA-Based Remote Laboratory in the Hybrid Cloud.” *Proceedings - 2017 International Conference on Cyberworlds, CW 2017 - in Cooperation with: Eurographics Association International Federation for Information Processing ACM SIGGRAPH* 2017-Janua:162–65.

Hrisko Joshua. 2020. “Capacitive Soil Moisture Sensor Calibration with Arduino — Maker Portal.” Retrieved April 6, 2021 (https://makersportal.com/blog/2020/5/26/capacitive-soil-moisture-calibration-with-arduino).

Junior, Manoel Jose, Orlewilson B. Maia, Horacio Oliveira, Eduardo Souto, and Raimundo Barreto. 2019. “Assistive Technology through Internet of Things and Edge Computing.” *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin* 2019-Septe:330–32.

Kumar, Jeya Amantha, and Paula Alexandra Silva. 2020. “Work-in-Progress: A Preliminary Study on Students’ Acceptance of Chatbots for Studio-Based Learning.” *IEEE Global Engineering Education Conference, EDUCON* 2020-April:1627–31.

Lafuente, Antonio Bustos. 2019. “Diseño y Control Domótico de Un Sistema de Riego Automático Para Un Huerto Urbano En El Hogar.” 8–8.

Mahmud, Istiak, and Nafis Almas Nafi. 2020. “An Approach of Cost-Effective Automatic Irrigation and Soil Testing System.” *ETCCE 2020 - International Conference on Emerging Technology in Computing, Communication and Electronics*.

Margret Sharmila, F., P. Suryaganesh, M. Abishek, and Ullas Benny. 2019. “Iot Based Smart Window Using Sensor Dht11.” *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019* (Icaccs):782–84.

Di Martino, Sergio, Luca Fiadone, Adriano Peron, Vincenzo Norman Vitale, and Alberto Riccabone. 2019. “Industrial Internet of Things: Persistence for Time Series with NoSQL Databases.” *Proceedings - 2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2019* 340–45.

Mecánica, Facultad DE, Presentado por, and Juan Diego Cruz Freire Darwin Vinicio Chimbo Chimbo. 2015. “Escuela Superior Politécnica De Chimborazo.”

Mir, Anam, and Ajitkumar Khachane. 2018. “Sensing Harmful Gases in Industries Using IOT and WSN.” *Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018* 18–20.

Muslih, Muhamad, Somantri, Dedi Supardi, Elpid Multipi, Yusup Maulana Nyaman, Aditya Rismawan, and Gunawansyah. 2019. “Developing Smart Workspace Based Iot with Artificial Intelligence Using Telegram Chatbot.” *Proceedings - 2018 4th International Conference on Computing, Engineering, and Design, ICCED 2018* 230–34.

Nageswara Rao, R., and B. Sridhar. 2018. “IoT Based Smart Crop-Field Monitoring and Automation Irrigation System.” *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018* (Icisc):478–83.

Nițulescu, Ioana-Victoria, and Adrian Korodi. 2020. “Supervisory Control and Data Acquisition Approach in Node-RED: Application and Discussions.” *IoT* 1(1):76–91.

openHAB. 2019a. “Rules | OpenHAB.” *OpenHAB Documentation*. Retrieved April 7, 2021 (https://www.openhab.org/docs/configuration/rules-dsl.html#textual-

rules).

openHAB. 2019b. “Things | OpenHAB.” *OpenHAB Documentation*. Retrieved April 7, 2021 (https://www.openhab.org/docs/concepts/things.html).

openHAB. 2021. “Pages - Overview Page | OpenHAB.” *OpenHAB Documentation*. Retrieved April 7, 2021 (https://www.openhab.org/docs/tutorial/auto_overview.html).

Perez, Raul. 2015. “Guía Para Realizar Un Buen Anáilisis Costo - Beneficio - Riesgo Para Un Proyecto De Erp Empresarial.” *TCA, Sofware* 15.

Pratyush Reddy, Kasara Sai, Y. Mohana Roopa, L. N. Kovvada Rajeev, and Narra Sai Nandan. 2020. “IoT Based Smart Agriculture Using Machine Learning.” *Proceedings of the 2nd International Conference on Inventive Research in Computing Applications, ICIRCA 2020* 130–34.

Puranik, Vaishali, Sharmila, Ankit Ranjan, and Anamika Kumari. 2019. “Automation in Agriculture and IoT.” *Proceedings - 2019 4th International Conference on Internet of Things: Smart Innovation and Usages, IoT-SIU 2019*.

Radi, Murtiningrum, Ngadisih, Fajar Siti Muzdrikah, M. Shohibun Nuha, and Fathi A. Rizqi. 2018. “Calibration of Capacitive Soil Moisture Sensor (SKU:SEN0193).” Pp. 1–6 in *Proceedings - 2018 4th International Conference on Science and Technology, ICST 2018*. IEEE.

Ramirez, Jhon, and Cesar Pedraza. 2017. “Performance Analysis of Communication Protocols for Internet of Things Platforms.” *2017 IEEE Colombian Conference on Communications and Computing, COLCOM 2017 - Proceedings*.

Salvi, Sanket, V. Geetha, and S. Sowmya Kamath. 2019. “Jamura: A Conversational Smart Home Assistant Built on Telegram and Google Dialogflow.” *IEEE Region 10 Annual International Conference, Proceedings/TENCON* 2019-Octob:1564–71.

Saxena, Srijan, Sahaj Jain, Deepak Arora, and Puneet Sharma. 2019. “Implications of MQTT Connectivity Protocol for Iot Based Device Automation Using Home Assistant and OpenHAB.” *Proceedings of the 2019 6th International Conference on Computing for Sustainable Global Development, INDIACom*

2019 475–80.

Selmani, A., M. Outanoute, M. Amini Alaoui, M. El Khayat, M. Guerbaoui, A. Ed-Dahhak, A. Lachhab, and B. Bouchikhi. 2018. “Multithreading Design for an Embedded Irrigation System Running on Solar Power.” *Proceedings of the 2018 International Conference on Optimization and Applications, ICOA 2018* 1–5.

Shilpa, A., V. Muneeswaran, and D. Devi Kala Rathinam. 2019. “A Precise and Autonomous Irrigation System for Agriculture: IoT Based Self Propelled Center Pivot Irrigation System.” *2019 5th International Conference on Advanced Computing and Communication Systems, ICACCS 2019* 533–38.

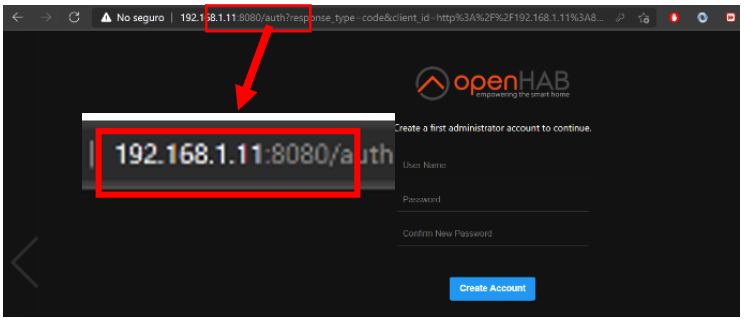
Veloo, Kesevan, Hayate Kojima, Shogo Takata, Masashi Nakamura, and Hironori Nakajo. 2019. “Interactive Cultivation System for the Future IoT-Based Agriculture.” *Proceedings - 2019 7th International Symposium on Computing and Networking Workshops, CANDARW 2019* 298–304.

Villena, Maiver, Victor Serrano, Daniel Hoyos, and Flavia Zutara. 2019. “Sistemas Embebidos Y Mqtt Para El Registro De Variables.” 7(1):129–39.

Widyawati, Dewi Kania, Agus Ambarwari, and Anung Wahyudi. 2020. “Design and Prototype Development of Internet of Things for Greenhouse Monitoring System.” *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2020* 389–93.

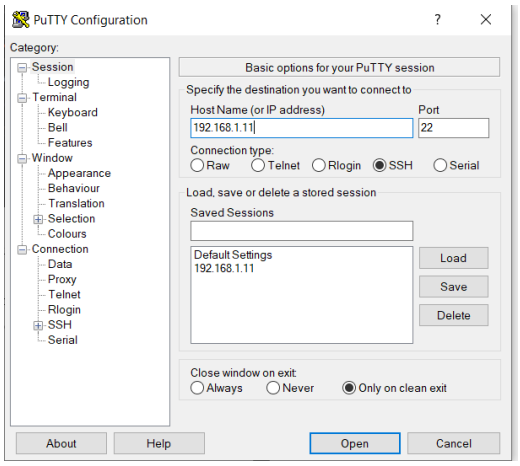
ANEXOS

Anexo 1: Presentación de la pantalla de inicio de Openhab.



Presentación de la pantalla de Openhab por medio de la IP establecida, Fuente: Openhab.

Anexo 2: Configuración con el programa PuTTY.



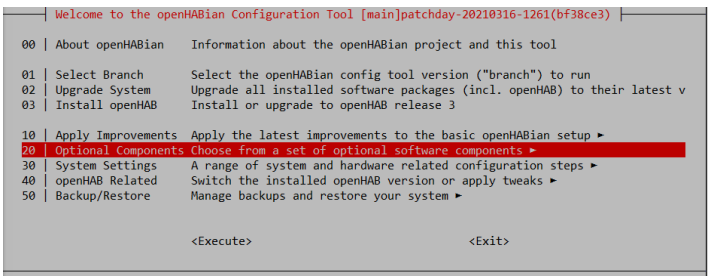
Por medio del programa PuTTY se ingresa a la Raspberry pi por medio de SSH, Fuente: PuTTY.

Anexo 3: Configuración para Openhabian.

```
openhabian@openhabian:~ $ sudo openhabian-config
```

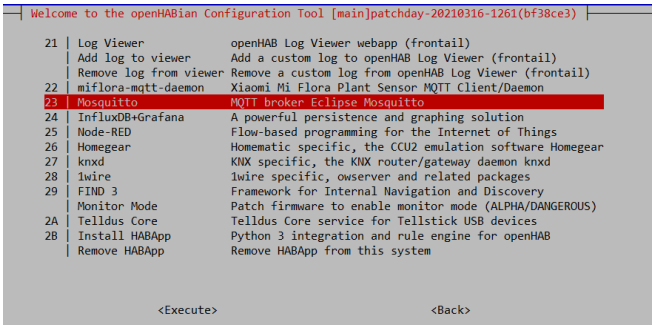
Comando para ingresar a la configuración de Openhabian, Fuente: Openhabian.

Anexo 4: Herramienta de Openhabian.



Herramienta de configuración para parámetros de Openhabian, Fuente: Openhabian.

Anexo 5: Instalación de Mosquitto.



Por medio de la Herramienta que ofrece Openhabian se debe instalar Mosquitto, Fuente: Openhabian.

Anexo 6: Comando de instalación de base de datos InfluxDB

```
openhabian@openhabian:~ $ sudo apt-get install influxdb
```

Fuente: Openhabian

Anexo 7: Comando para que InfluxDB funcione como servicio.

```
openhabian@openhabian:~ $ sudo systemctl unmask influxdb.service
```

Fuente: Openhabian.

Anexo 8: Comando para iniciar InfluxDB.

```
openhabian@openhabian:~ $ sudo systemctl start influxdb
```

Fuente: Openhabian.

Anexo 9: Comando de ingreso a fichero de InfluxDB.

```
openhabian@openhabian:~ $ sudo nano /etc/influxdb/influxdb.conf
```

Fuente: Openhabian.

Anexo 10: Autentificación http.

```
[http]
  # Determines whether HTTP endpoint is enabled.
  enabled = true

  # Determines whether the Flux query endpoint is enabled.
  # flux-enabled = false

  # Determines whether the Flux query logging is enabled.
  # flux-log-enabled = false

  # The bind address used by the HTTP service.
  bind-address = ":8086"

  # Determines whether user authentication is enabled over HTTP/HTTPS.
  auth-enabled = true
```

Autentificación http en el fichero de configuración de InfluxDB, Fuente: Openhabian

Anexo 11: Configuración de base de datos.

```
openhabian@openhabian:~ $ influx
Connected to http://localhost:8086 version 1.8.4
InfluxDB shell version: 1.8.4
> create database openhab_db
> CREATE USER admin WITH PASSWORD 'titulacion' WITH ALL PRIVILEGES
> CREATE USER openhab WITH PASSWORD 'titulacion'
```

Configuración de base de datos con contraseña y usuario, Fuente: Openhabian

Anexo 12: Comando de asignación de permisos.

```
> GRANT ALL ON openhab_db TO openhab
```

Fuente: Openhabian

Anexo 13: Configuración de vinculación de Openhab con InfluxDB.

```
openhabian@openhabian:~ $ sudo nano /etc/openhab/services/influxdb.cfg
```

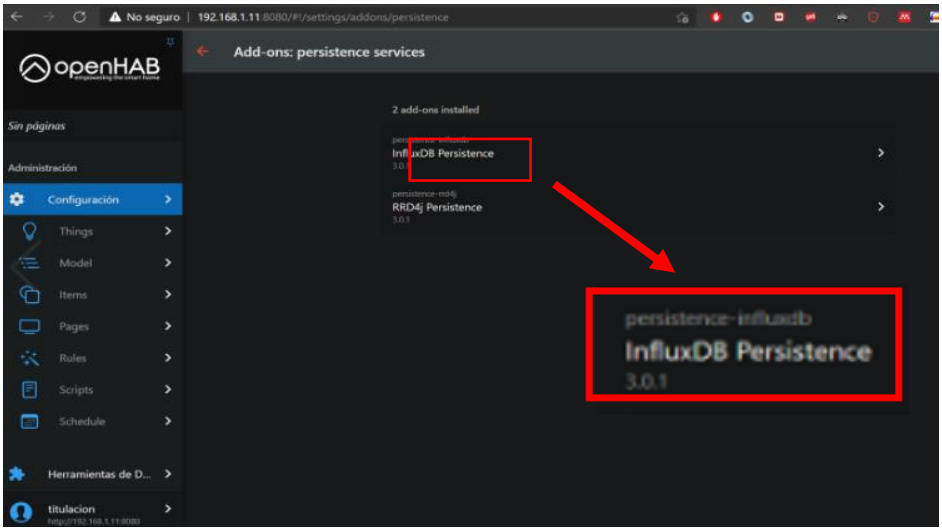
Fuente: Openhabian

Anexo 14: Ingreso de usuario y contraseña.

```
 GNU nano 3.2                    /etc/openhab/services/influxdb.cfg

# The database URL, e.g. http://127.0.0.1:8086 or https://127.0.0.1:8084 .
# Defaults to: http://127.0.0.1:8086
url=http://192.168.1.11:8086

# The name of the database user, e.g. openhab.
# Defaults to: openhab
user=openhab

# The password of the database user.
password=titulacion

# The name of the database, e.g. openhab.
# Defaults to: openhab
db=openhab_db
```

Se ingresa el usuario, contraseña y se especifica la base de datos que se utilizará, Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 15: Selección de base de datos.



Selección de base de datos que se utilizará en este proyecto, Fuente: Openhabian.

Anexo 16: Ingreso mediante SSH.

```
openhabian@openhabian:~ $ sudo nano /etc/lib/openhab/openhabcloud/secret
```

Ingreso al fichero secret para la obtención de la clave, Fuente: Openhabian.

Anexo 17: Comando de ingreso al fichero.

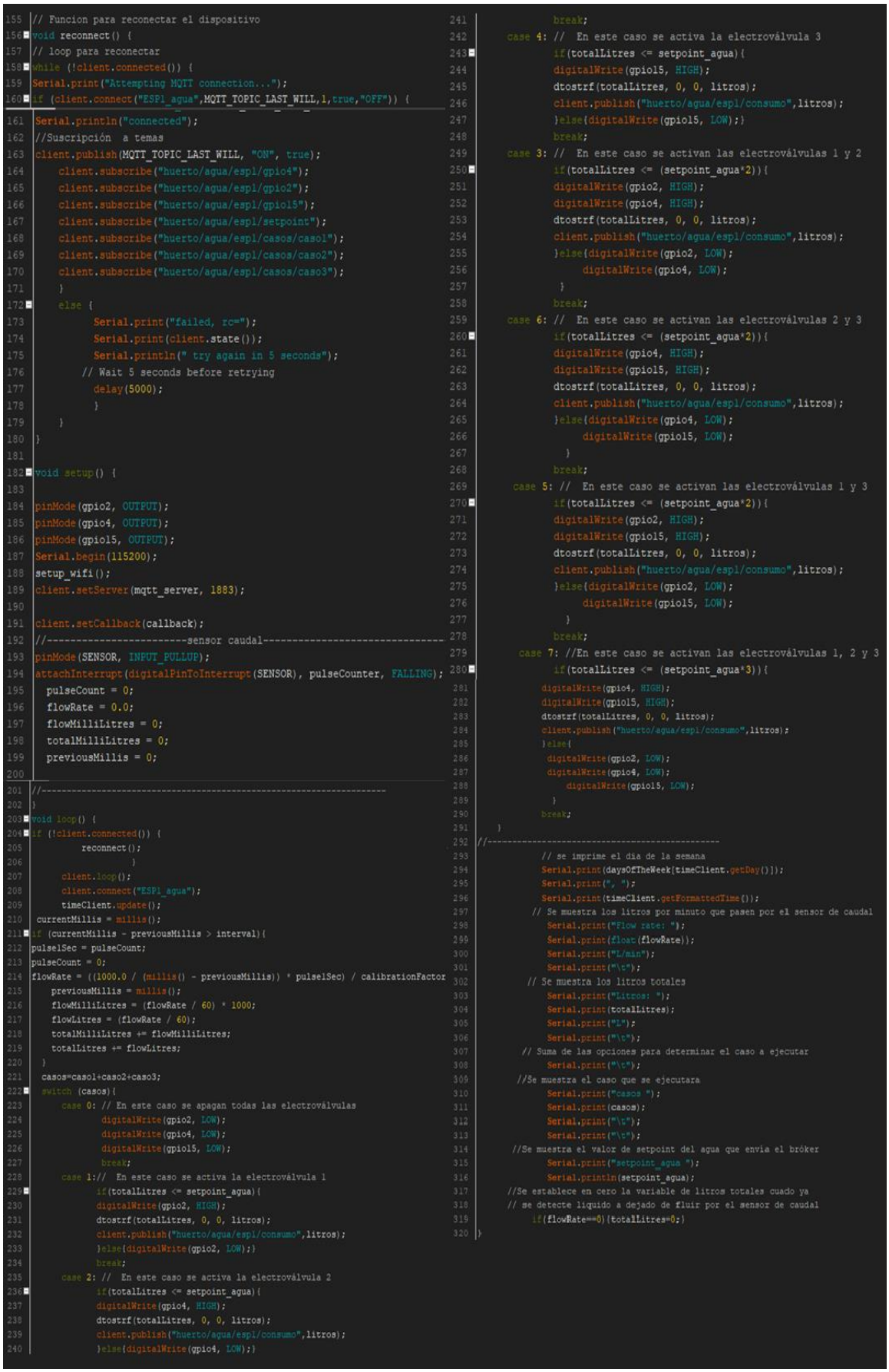
```
openhabian@openhabian:~ $ nano /var/lib/openhab/uuid
```

Fuente: Openhabian.

Anexo 18: Programación del Módulo 1

```
//------------Librerias------------------
#include <WiFi.h>
#include <WiFiUdp.h>
#include <PubSubClient.h>
//----------------------------------------
#include <NTPClient.h>
const long utcOffsetInSeconds = -18000;
char daysOfTheWeek[7][12] = {"Domingo", "Lunes", "Martes",
                "Miercoles","Jueves", "Viernes", "Sabado"};
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP,
         "south-america.pool.ntp.org", utcOffsetInSeconds);
//----------------------------------------
int setpoint_agua =0;
int casos=0 ;
int caso1 =0;
int caso2 =0;
int caso3 =0;
//---------- declaracion de leds
#define gpio2  2  //Electroválvula 1
#define gpio4  4 //Electroválvula 2
#define gpio15 15 //Electroválvula 3
#define MQTT_TOPIC_LAST_WILL "huerto/agua/esp1/lwt"
//-------------medidor de caudal---------------
#define SENSOR  14
static char litros[7];
long currentMillis = 0;
long previousMillis = 0;
int interval = 1000;
boolean ledState = LOW;
float calibrationFactor = 4.5;
volatile byte pulseCount;
byte pulse1Sec = 0;
float flowRate;
unsigned long flowMilliLitres;
unsigned int totalMilliLitres;
float flowLitres;
float totalLitres;
void IRAM_ATTR pulseCounter()
{
  pulseCount++;
}
//************Configuracion de IP estatica **************************
IPAddress ip(192,168,1,31);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
//********SSID y conrtraseña del router*****************************
const char* ssid = "CELERITY_SOFY";
const char* password = "$%Thepolice";
//+++++++++++++++++ direccion de raspberry  MQTT broker+++++++++++++++
const char* mqtt_server = "192.168.1.11";
const char* mqtt_username = "mqtt";
const char* mqtt_password = "12345678";
//-----------------------------------------------------------------
//Inicio de cliente. Se debe cambiar el nombre de "espClient" si
//se tiene diferentes ESP en el sistema
//-------------------------------------------------
WiFiClient esp1Cliente;
//----------
PubSubClient client(esp1Cliente);
//--------- Timers variables auxiliares
long now = millis();
long lastMeasure = 0;
//-------NO CAMBIAR ESTA FUNCION ES PARA CONEXION CON EL ROUTER!!!---
           void setup_wifi() {
           delay(10);
           // We start by connecting to a WiFi network
           Serial.println();
           Serial.print("Connecting to ");
           Serial.println(ssid);
           WiFi.begin(ssid, password);
           while (WiFi.status() != WL_CONNECTED) {
           delay(500);
           Serial.print(".");
           }
           timeClient.begin();
           Serial.println("");
           Serial.print("WiFi connected - ESP IP address: ");
           Serial.println(WiFi.localIP());
           }
//-----------------------------------------------------------------------
void callback(String topic, byte* message, unsigned int length) {
Serial.print("Message arrived on topic: ");
Serial.print(topic);
Serial.print(". Message: ");
String messageTemp;
for (int i = 0; i < length; i++) {
Serial.print((char)message[i]);
messageTemp += (char)message[i];
}
Serial.println();
  if(topic=="huerto/agua/esp1/setpoint"){
     setpoint_agua = String(messageTemp).toInt();
    Serial.print("setpoint_agua= ");
    Serial.println(setpoint_agua);
//-----------------------------------------------
    }
    if(topic=="huerto/agua/esp1/casos/caso1"){
     caso1 =  String(messageTemp).toInt();
    Serial.print("casos1= ");
    Serial.println(caso1);
    }
    if(topic=="huerto/agua/esp1/casos/caso2"){
     caso2 =  String(messageTemp).toInt();
    Serial.print("casos2= ");
    Serial.println(caso2);
    }
    if(topic=="huerto/agua/esp1/casos/caso3"){
     caso3 =  String(messageTemp).toInt();
    Serial.print("casos3= ");
    Serial.println(caso3);
    }
  //-----------------------------------------
  if(topic=="huerto/agua/esp1/gpio4"){
     Serial.print("Changing GPIO 4 to ");
    if(messageTemp == "1"){
      digitalWrite(gpio4, HIGH);
      client.publish("huerto/agua/esp1/gpio4/estado", "ON",1);
      Serial.print("ON");
      }
        else if(messageTemp == "0"){
        digitalWrite(gpio4, LOW);
        client.publish("huerto/agua/esp1/gpio4/estado", "OFF",1);
        Serial.print("OFF");
              }
  }
      if(topic=="huerto/agua/esp1/gpio2"){
         Serial.print("Changing GPIO 2 to ");
            if(messageTemp == "1"){
               digitalWrite(gpio2, HIGH);
               client.publish("huerto/agua/esp1/gpio2/estado", "ON");
               Serial.print("ON");
             }
               else if(messageTemp == "0"){
               digitalWrite(gpio2, LOW);
               client.publish("huerto/agua/esp1/gpio2/estado", "OFF");
               Serial.print("OFF");
                     }
           }
      if(topic=="huerto/agua/esp1/gpio15"){
         Serial.print("Changing GPIO 15 to ");
            if(messageTemp == "1"){
               digitalWrite(gpio15, HIGH);
               client.publish("huerto/agua/esp1/gpio15/estado", "ON");
               Serial.print("ON");
             }
               else if(messageTemp == "0"){
               digitalWrite(gpio15, LOW);
               client.publish("huerto/agua/esp1/gpio15/estado", "OFF");
               Serial.print("OFF");
                     }
           }
Serial.println();
}
```

Elaborado por: Estefania Acosta & Carlos Cuaical.



```
// Funcion para reconectar el dispositivo
void reconnect() {
// loop para reconectar
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
if (client.connect("ESP1_agua",MQTT_TOPIC_LAST_WILL,1,true,"OFF")) {
Serial.println("connected");
//Suscripción  a temas
client.publish(MQTT_TOPIC_LAST_WILL, "ON", true);
    client.subscribe("huerto/agua/esp1/gpio4");
    client.subscribe("huerto/agua/esp1/gpio2");
    client.subscribe("huerto/agua/esp1/gpio15");
    client.subscribe("huerto/agua/esp1/setpoint");
    client.subscribe("huerto/agua/esp1/casos/caso1");
    client.subscribe("huerto/agua/esp1/casos/caso2");
    client.subscribe("huerto/agua/esp1/casos/caso3");
    }
    else {
         Serial.print("failed, rc=");
         Serial.print(client.state());
         Serial.println(" try again in 5 seconds");
       // Wait 5 seconds before retrying
         delay(5000);
         }
    }
}

void setup() {

pinMode(gpio2, OUTPUT);
pinMode(gpio4, OUTPUT);
pinMode(gpio15, OUTPUT);
Serial.begin(115200);
setup_wifi();
client.setServer(mqtt_server, 1883);

client.setCallback(callback);
//-----------------------sensor caudal-------------------------------
pinMode(SENSOR, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(SENSOR), pulseCounter, FALLING);
  pulseCount = 0;
  flowRate = 0.0;
  flowMilliLitres = 0;
  totalMilliLitres = 0;
  previousMillis = 0;

//---------------------------------------------------------------
}
void loop() {
if (!client.connected()) {
        reconnect();
                }
    client.loop();
    client.connect("ESP1_agua");
    timeClient.update();
 currentMillis = millis();
if (currentMillis - previousMillis > interval){
pulse1Sec = pulseCount;
pulseCount = 0;
flowRate = ((1000.0 / (millis() - previousMillis)) * pulse1Sec) / calibrationFactor
   previousMillis = millis();
   flowMilliLitres = (flowRate / 60) * 1000;
   flowLitres = (flowRate / 60);
   totalMilliLitres += flowMilliLitres;
   totalLitres += flowLitres;
  }
  casos=caso1+caso2+caso3;
  switch (casos){
     case 0: // En este caso se apagan todas las electroválvulas
            digitalWrite(gpio2, LOW);
            digitalWrite(gpio4, LOW);
            digitalWrite(gpio15, LOW);
            break;
     case 1://  En este caso se activa la electroválvula 1
           if(totalLitres <= setpoint_agua){
           digitalWrite(gpio2, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio2, LOW);}
           break;
     case 2: //  En este caso se activa la electroválvula 2
           if(totalLitres <= setpoint_agua){
           digitalWrite(gpio4, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio4, LOW);}
           break;
     case 4: //  En este caso se activa la electroválvula 3
           if(totalLitres <= setpoint_agua){
           digitalWrite(gpio15, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio15, LOW);}
           break;
     case 3: //  En este caso se activan las electroválvulas 1 y 2
           if(totalLitres <= (setpoint_agua*2)){
           digitalWrite(gpio2, HIGH);
           digitalWrite(gpio4, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio2, LOW);
                digitalWrite(gpio4, LOW);
            }
           break;
     case 6: //  En este caso se activan las electroválvulas 2 y 3
           if(totalLitres <= (setpoint_agua*2)){
           digitalWrite(gpio4, HIGH);
           digitalWrite(gpio15, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio4, LOW);
                digitalWrite(gpio15, LOW);
             }
           break;
      case 5: //  En este caso se activan las electroválvulas 1 y 3
           if(totalLitres <= (setpoint_agua*2)){
           digitalWrite(gpio2, HIGH);
           digitalWrite(gpio15, HIGH);
           dtostrf(totalLitres, 0, 0, litros);
           client.publish("huerto/agua/esp1/consumo",litros);
           }else{digitalWrite(gpio2, LOW);
                digitalWrite(gpio15, LOW);
             }
           break;
       case 7: //En este caso se activan las electroválvulas 1, 2 y 3
           if(totalLitres <= (setpoint_agua*3)){
          digitalWrite(gpio4, HIGH);
          digitalWrite(gpio15, HIGH);
          dtostrf(totalLitres, 0, 0, litros);
          client.publish("huerto/agua/esp1/consumo",litros);
          }else{
           digitalWrite(gpio2, LOW);
           digitalWrite(gpio4, LOW);
              digitalWrite(gpio15, LOW);
            }
          break;
    }
//---------------------------------------------
          // se imprime el dia de la semana
          Serial.print(daysOfTheWeek[timeClient.getDay()]);
          Serial.print(", ");
          Serial.print(timeClient.getFormattedTime());
        // Se muestra los litros por minuto que pasen por el sensor de caudal
          Serial.print("Flow rate: ");
          Serial.print(float(flowRate));
          Serial.print("L/min");
          Serial.print("\t");
       // Se muestra los litros totales
          Serial.print("Litros: ");
          Serial.print(totalLitres);
          Serial.print("L");
          Serial.print("\t");
      // Suma de las opciones para determinar el caso a ejecutar
          Serial.print("\t");
      //Se muestra el caso que se ejecutara
          Serial.print("casos ");
          Serial.print(casos);
          Serial.print("\t");
          Serial.print("\t");
     //Se muestra el valor de setpoint del agua que envia el bróker
          Serial.print("setpoint_agua ");
          Serial.println(setpoint_agua);
    //Se establece en cero la variable de litros totales cuado ya
    // se detecte liquido a dejado de fluir por el sensor de caudal
        if(flowRate==0){totalLitres=0;}
}
```

Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 19: Parámetros iniciales para la calibración.

Sensor 1	
Masa-recipiente (Mr)	9,79 [gr]
Densidad del agua	0,997 [kg/L]
Volumen	200 [ml]
Densidad del suelo	0,92 [g/ml]
Masa del suelo seco (Ms)	184 [gr]
voltios al aire libre[v]	2,88

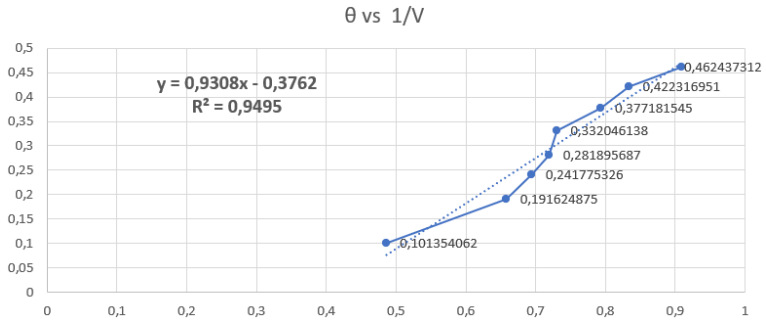
Parámetros iniciales para la calibración de sensores capacitivos de humedad de suelo, Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 20: Valores del proceso de calibración.

Muestra #	Msh+Mr [gr]	[v]	1 / [v]	Msh-Mr	θ
1	214	2,06	0,48543689	204,21	0,10135406
2	232	1,52	0,65789474	222,21	0,19162487
3	242	1,44	0,69444444	232,21	0,24177533
4	250	1,39	0,71942446	240,21	0,28189569
5	260	1,37	0,72992701	250,21	0,33204614
6	269	1,26	0,79365079	259,21	0,37718154
7	278	1,2	0,83333333	268,21	0,42231695
8	286	1,1	0,90909091	276,21	0,46243731

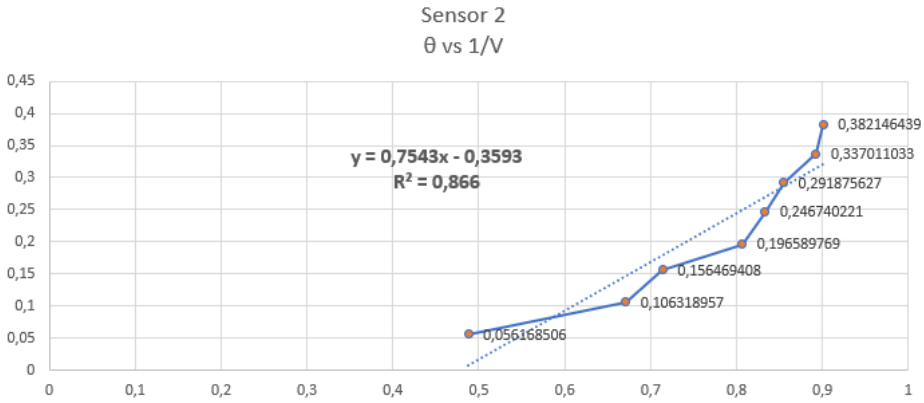
Valores del proceso de calibracioón por medio del método de la gravimetría, Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 21: Gráfica para la función de transferencia del sensor 1.



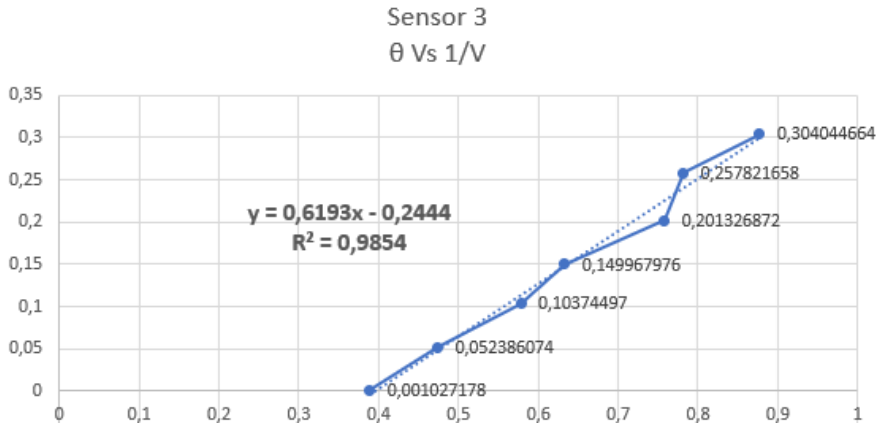
Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 22: Gráfica para la función de transferencia del sensor 2.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 23: Gráfica para la función de transferencia del sensor 3.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 24: Lecturas de los sensores calibrados.

```
Voltage_1: 1.51 θ_1= 0.24 %_1 = 55.2257 Voltage_2: 1.30 θ_2 = 0.22 %_2 = 57.9051 Voltage_3: 1.47   θ_3 = 0.1757  %_3= 55.4827
Voltage_1: 1.51 θ_1= 0.24 %_1 = 55.2141 Voltage_2: 1.31 θ_2 = 0.22 %_2 = 57.1478 Voltage_3: 1.47   θ_3 = 0.1761  %_3= 55.6178
Voltage_1: 1.52 θ_1= 0.24 %_1 = 55.0749 Voltage_2: 1.31 θ_2 = 0.22 %_2 = 56.9163 Voltage_3: 1.48   θ_3 = 0.1754  %_3= 55.3815
Voltage_1: 1.52 θ_1= 0.24 %_1 = 55.2025 Voltage_2: 1.31 θ_2 = 0.22 %_2 = 56.7575 Voltage_3: 1.47   θ_3 = 0.1757  %_3= 55.4827
Voltage_1: 1.51 θ_1= 0.24 %_1 = 55.2141 Voltage_2: 1.31 θ_2 = 0.22 %_2 = 57.3945 Voltage_3: 1.47   θ_3 = 0.1771  %_3= 55.9228
```

Lectura de los sensores calibrados, Elaborado por: Estefania Acosta & Carlos Cuaical.

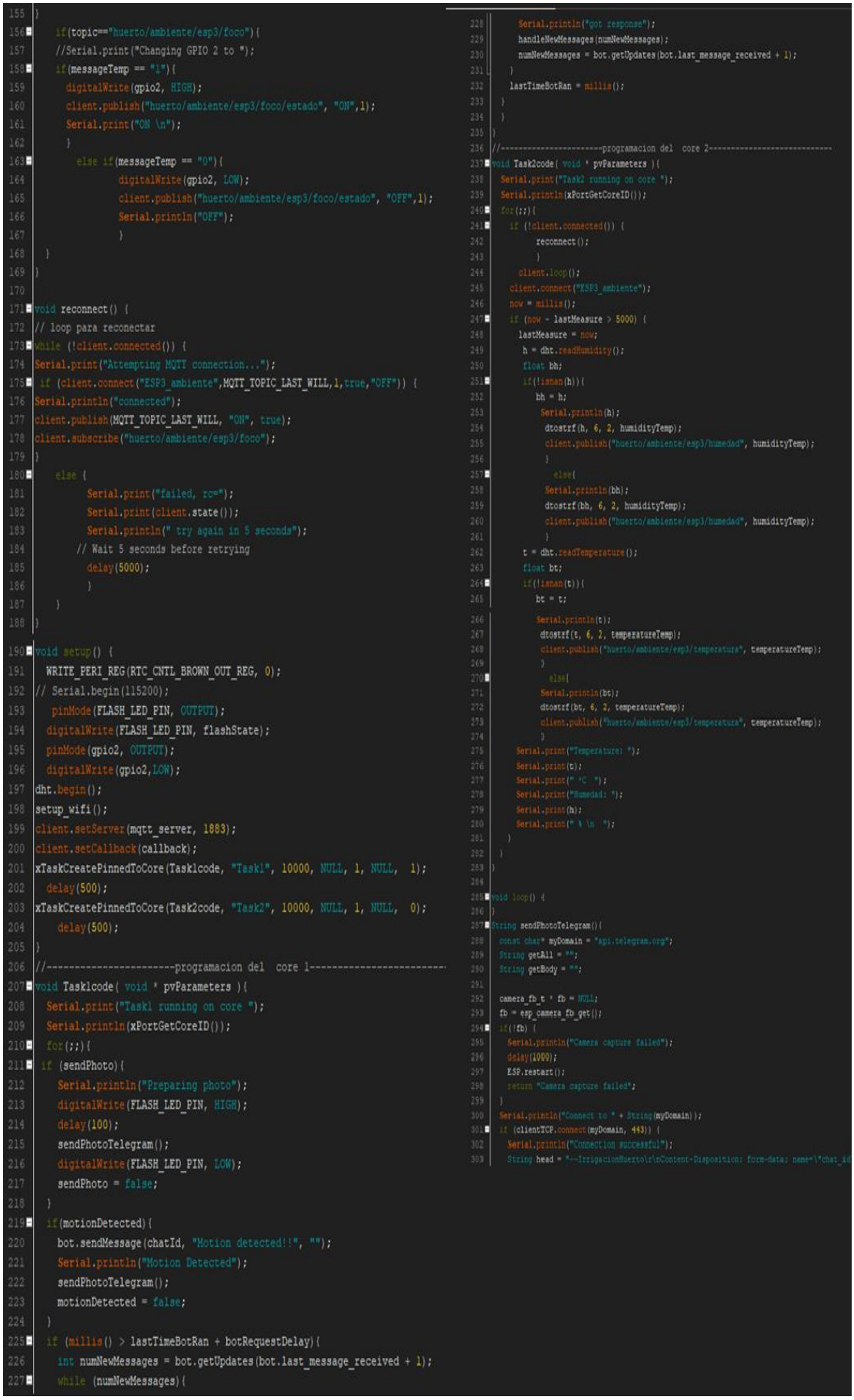
Anexo 25: Programación del Módulo 2

```
//------libreria para obtener la moda-----
#include <Average.h>
Average<float> ave(50);
Average<float> ave1(50);
Average<float> ave2(50);
//********ADS1115***************
#include <Wire.h>
#include <Adafruit_ADS1015.h>
Adafruit_ADS1115 ads(0x48);
float factorEscala=0.125F;
float Voltage = 0.0;
float Voltage1 = 0.0;
float Voltage2 = 0.0;
float vol_water_cont;
//-------------------------------
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
#define MQTT_TOPIC_LAST_WILL "huerto/sensores/esp2/lwt"
#include <PubSubClient.h>
//---------hora-------------
#include <NTPClient.h>
#include <WiFiUdp.h>
//--------- Timers variables auxiliares
long now = millis();
long lastMeasure = 0;
//-----------------------------------
int16_t adc0;
int16_t adc1;
int16_t adc2;
int i=0;
static char voltaje_sensor[7];
static char volumen_agua_cont0 [7];
static char porcentaje_0 [7];
static char voltaje1_sensor[7];
static char volumen_agua_cont1[7];
static char porcentaje_1 [7];
static char voltaje2_sensor[7];
static char volumen_agua_cont2[7];
static char porcentaje_2 [7];
//--------------ip estatica -------------
IPAddress ip(192,168,1,32);
//IPAddress ip(192,168,1,32);
IPAddress gateway(192,168,1,1);
IPAddress subnet(255,255,255,0);
//**************SSID y contraseña-------------------------
const char* ssid = "CELERITY_SOFY";
const char* password = "$%Thepolice";
//++++++++direccion de raspberry  MQTT broker+++++++++++++++++
const char* mqtt_server = "192.168.1.11";
const char* mqtt_username = "mqtt";
const char* mqtt_password = "12345678";
//-------------------------------------------------------------
//Inicio de cliente. Se debe cambiar el nombre de "espClient" si
//se tiene diferentes ESP en el sistema
//-----------------------------------------------
WiFiClient esp2Cliente;
//----------
PubSubClient client(esp2Cliente);
//-------NO CAMBIAR ESTA FUNCION ES PARA CONEXION CON EL ROUTER!!!-
          void setup_wifi() {
          delay(10);
          Serial.println();
          Serial.print("Connecting to ");
          Serial.println(ssid);
          WiFi.begin(ssid, password);
          while (WiFi.status() != WL_CONNECTED) {
          delay(500);
          Serial.print(".");
          }
          Serial.println("");
          Serial.print("WiFi connected - ESP IP address: ");
          Serial.println(WiFi.localIP());
          }
//-------------------------------------------------------------
void callback(String topic, byte* message, unsigned int length) {
Serial.print("Message arrived on topic: ");
Serial.print(topic);
Serial.print(". Message: ");
String messageTemp;
for (int i = 0; i < length; i++) {
Serial.print((char)message[i]);
messageTemp += (char)message[i];
}
Serial.println();
}
void reconnect() {
// loop para reconectar
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
      if (client.connect("ESP2_sensores",MQTT_TOPIC_LAST_WILL,1,true,"OFF")) {
Serial.println("connected");
client.publish(MQTT_TOPIC_LAST_WILL, "ON", true);
    }
    else {
          Serial.print("failed, rc=");
          Serial.print(client.state());
          Serial.println(" try again in 5 seconds");
          delay(5000);
          }
    }
}
void setup() {
Serial.begin(115200);
//--ADS1115--
ads.setGain(GAIN_ONE);
ads.begin();
//-----------
setup_wifi();
client.setServer(mqtt_server, 1883);
client.setCallback(callback);
}
//////////////////////----------------------------
void loop() {
      if (!client.connected()) {
          reconnect();
          }
          client.loop();
          client.connect("ESP2_sensores");
           now = millis();

 //----inicio de bucle para tomar lectura de 50 datos-------------
 for (int i = 0; i < 50; i++) {
adc0 = ads.readADC_SingleEnded(0);
adc1 = ads.readADC_SingleEnded(1);
adc2 = ads.readADC_SingleEnded(2);
Voltage = (adc0 * factorEscala)/1000.0;
Serial.print("Voltage_1: ");
Serial.print(Voltage, 2);
dtostrf(Voltage, 6, 1, voltaje_sensor);
//---------funcion de transferencia del sensor 1-------------------
vol_water_cont = ((1.0/Voltage)*0.938)+(-0.3762);
Serial.print(" θ_1= ");Serial.print(vol_water_cont);
float porcentaje= vol_water_cont*100.0/0.40;
Serial.print(" %_1 = ");Serial.print(porcentaje,4);
dtostrf(porcentaje, 0, 0, porcentaje_0);
dtostrf(vol_water_cont, 6, 2, volumen_agua_cont0);
//-----------------------------------------------
ave.push(porcentaje);
//-----------------------------------adc1-------------------------
Voltage1 = (adc1 * factorEscala)/1000.0;
Serial.print(" Voltage_2: ");
Serial.print(Voltage1, 2);
dtostrf(Voltage1, 6, 1, voltaje1_sensor);
float vol_water_cont1 = ((1.0/Voltage1)*0.7543)-0.3593;
dtostrf(vol_water_cont1, 6, 2, volumen_agua_cont1);
 Serial.print(" θ_2 = ");Serial.print(vol_water_cont1);
 float porcentaje1= vol_water_cont1*100.0/0.38;
 avel.push(porcentaje1);
 Serial.print(" %_2 = ");Serial.print(porcentaje1,4);
 dtostrf(porcentaje1, 0, 0, porcentaje_1);
//-----------------------------------adc2-------------------------
Voltage2 = (adc2 * factorEscala)/1000.0;
Serial.print(" Voltage_3: ");
Serial.print(Voltage2, 2);
dtostrf(Voltage2, 6, 1, voltaje2_sensor);
float vol_water_cont2 = ((1.0/Voltage2)*0.6193-0.2444);
dtostrf(vol_water_cont2, 6, 2, volumen_agua_cont2);
 Serial.print("   θ_3 = ");Serial.print(vol_water_cont2,4);
 float porcentaje2= vol_water_cont2*100.0/0.3167;
 ave2.push(porcentaje2);
 Serial.print("  %_3= ");Serial.println(porcentaje2,4);
  dtostrf(porcentaje2, 0, 0, porcentaje_2);
  }
  Serial.print("Mode:   "); Serial.println(ave.mode());
  Serial.print("Mode1:   "); Serial.println(avel.mode());
  Serial.print("Mode2:   "); Serial.println(ave2.mode());
 //----Almacenamiento de los valores de la moda de cada sensor
  float moda0 =ave.mode();
   float moda1 =avel.mode();
    float moda2 =ave2.mode();
 //--- Transformación de valores de float a string--------------
  dtostrf(moda0, 0, 0, porcentaje_0);
  dtostrf(moda1, 0, 0, porcentaje_1);
  dtostrf(moda2, 0, 0, porcentaje_2);
 //---------------Envio de datos al broker cada 2 segundos----------
  if (now - lastMeasure > 2000) {
              lastMeasure = now;
client.publish("huerto/sensores/esp2/voltaje_sensor", voltaje_sensor);
client.publish("huerto/sensores/esp2/volumen/volumen_agua_cont0", volumen_agua_cont0);
client.publish("huerto/sensores/esp2/porcentaje/porcentaje_0",porcentaje_0 );
client.publish("huerto/sensores/esp2/voltaje1_sensor", voltaje1_sensor);
client.publish("huerto/sensores/esp2/volumen/volumen_agua_cont1", volumen_agua_cont1);
client.publish("huerto/sensores/esp2/porcentaje/porcentaje_1",porcentaje_1 );
client.publish("huerto/sensores/esp2/voltaje2_sensor", voltaje2_sensor);
client.publish("huerto/sensores/esp2/volumen/volumen_agua_cont2", volumen_agua_cont2);
client.publish("huerto/sensores/esp2/porcentaje/porcentaje_2",porcentaje_2 );
    }
}
```

Elaborado por: Estefania Acosta & Carlos Cuaical.

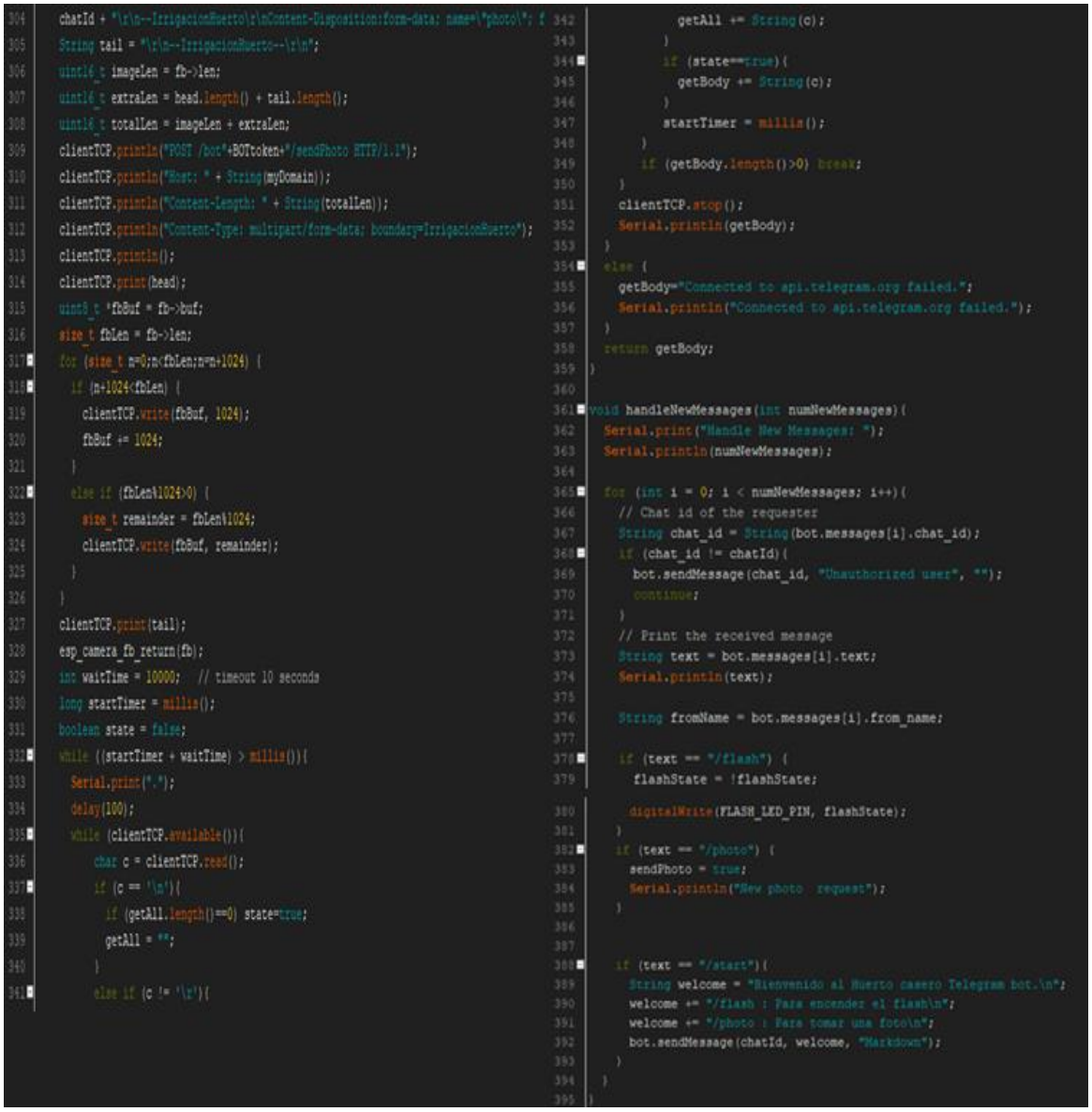
Anexo 26: Programación del módulo 3

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiServer.h>
//---------------------------------------
#include <WiFiClientSecure.h>
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>
#include <Wire.h>
//---------------------------------------
#define MQTT_TOPIC_LAST_WILL "huerto/ambiente/esp3/lwt"
#include <PubSubClient.h>
#include "DHT.h"
#define DHTPIN 14
#define DHTTYPE DHT11
float h;//variable de humedad
float t;//variable de temperatura
const int gpio2 = 2;
DHT dht(DHTPIN, DHTTYPE);
//--------- Timers variables auxiliares
long now = millis();
long lastMeasure = 0;
static char humidityTemp[7];
static char temperatureTemp[7];
//-----------------------------------------------------------
const char* ssid = "CELERITY_SOFY";
const char* password = "$%Thepolice";
//-----------------------------------------------------------------
String BOTtoken = "1730827677:AAFMnFwZYdjMJgQA18V1Gfk78zLk5JLTxz4";
String chatId ="-590161285";
bool sendPhoto = false;
WiFiClientSecure clientTCP;
UniversalTelegramBot bot(BOTtoken, clientTCP);
//CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22
#define FLASH_LED_PIN 4
bool flashState = LOW;
// Motion Sensor
bool motionDetected = false;
int botRequestDelay = 1000;   // mean time between scan messages
long lastTimeBotRan;     // last time messages' scan has been done
void handleNewMessages(int numNewMessages);
String sendPhotoTelegram();
// Indicates when motion is detected
static void IRAM_ATTR detectsMovement(void * arg){
  //Serial.println("MOTION DETECTED!!!");
  motionDetected = true;
}
//-----------------------------------------------------------------
//+++++++++++++++ direccion de raspberry  MQTT broker++++++++++++++++
const char* mqtt_server = "192.168.1.11";
const char* mqtt_username = "mqtt";
const char* mqtt_password = "12345678";
//-----------------------------------------------------------------
WiFiClient esp3Cliente;
PubSubClient client(esp3Cliente);
//*****************************************************************
//-------NO CAMBIAR ESTA FUNCION ES PARA CONEXION CON EL ROUTER!!!---
          void setup_wifi() {
          delay(10);
          // We start by connecting to a WiFi network
          Serial.println();
          Serial.print("Connecting to ");
          Serial.println(ssid);
          WiFi.begin(ssid, password);
          while (WiFi.status() != WL_CONNECTED) {
          delay(500);
          Serial.print(".");
          }
//-----------------------------------------------------------------------
 camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  //init with high specs to pre-allocate larger buffers
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;  //0-63 lower number means higher quality
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;  //0-63 lower number means higher quality
    config.fb_count = 1;
  }
  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    delay(1000);
    ESP.restart();
  }
  // Drop down frame size for higher initial frame rate
  sensor_t * s = esp_camera_sensor_get();
  // s->set_framesize(s, FRAMESIZE_VGA);  // UXGA|SXGA|XGA|SVGA|VGA|CIF|QVGA|HQVGA|QQVGA
  s->set_framesize(s, FRAMESIZE_SVGA);
  // PIR Motion Sensor mode INPUT_PULLUP
  //err = gpio_install_isr_service(0);
  err = gpio_isr_handler_add(GPIO_NUM_13, &detectsMovement, (void *) 13);
  if (err != ESP_OK){
    Serial.printf("handler add failed with error 0x%x \r\n", err);
  }
  err = gpio_set_intr_type(GPIO_NUM_13, GPIO_INTR_POSEDGE);
  if (err != ESP_OK){
    Serial.printf("set intr type failed with error 0x%x \r\n", err);
  }
//-----------------------------------------------------------------------
          Serial.println("");
          Serial.print("WiFi connected - ESP IP address: ");
          Serial.println(WiFi.localIP());
          }
//---------------/-----------------------------------------------------
void callback(String topic, byte* message, unsigned int length) {
//Serial.print("Message arrived on topic: ");
//Serial.print(topic);
//Serial.print(", Message: ");
String messageTemp;

for (int i = 0; i < length; i++) {
//Serial.print((char)message[i]);
messageTemp += (char)message[i];
```



```
}
    if(topic=="huerto/ambiente/esp3/foco"){
    //Serial.print("Changing GPIO 2 to ");
    if(messageTemp == "1"){
      digitalWrite(gpio2, HIGH);
      client.publish("huerto/ambiente/esp3/foco/estado", "ON",1);
      Serial.print("ON \n");
      }
        else if(messageTemp == "0"){
              digitalWrite(gpio2, LOW);
              client.publish("huerto/ambiente/esp3/foco/estado", "OFF",1);
              Serial.println("OFF");
              }
  }
}

void reconnect() {
// loop para reconectar
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
 if (client.connect("ESP3_ambiente",MQTT_TOPIC_LAST_WILL,1,true,"OFF")) {
Serial.println("connected");
client.publish(MQTT_TOPIC_LAST_WILL, "ON", true);
client.subscribe("huerto/ambiente/esp3/foco");
}
    else {
          Serial.print("failed, rc=");
          Serial.print(client.state());
          Serial.println(" try again in 5 seconds");
        // Wait 5 seconds before retrying
          delay(5000);
          }
    }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);
// Serial.begin(115200);
   pinMode(FLASH_LED_PIN, OUTPUT);
  digitalWrite(FLASH_LED_PIN, flashState);
  pinMode(gpio2, OUTPUT);
  digitalWrite(gpio2,LOW);
dht.begin();
setup_wifi();
client.setServer(mqtt_server, 1883);
client.setCallback(callback);
xTaskCreatePinnedToCore(Task1code, "Task1", 10000, NULL, 1, NULL,  1);
  delay(500);
xTaskCreatePinnedToCore(Task2code, "Task2", 10000, NULL, 1, NULL,  0);
    delay(500);
}
//----------------------programacion del  core 1-----------------------
void Task1code( void * pvParameters ){
  Serial.print("Task1 running on core ");
  Serial.println(xPortGetCoreID());
  for(;;){
  if (sendPhoto){
    Serial.println("Preparing photo");
    digitalWrite(FLASH_LED_PIN, HIGH);
    delay(100);
    sendPhotoTelegram();
    digitalWrite(FLASH_LED_PIN, LOW);
    sendPhoto = false;
  }
  if(motionDetected){
    bot.sendMessage(chatId, "Motion detected!!", "");
    Serial.println("Motion Detected");
    sendPhotoTelegram();
    motionDetected = false;
  }
  if (millis() > lastTimeBotRan + botRequestDelay){
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    while (numNewMessages){
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }
  }
}
//----------------------programacion del  core 2-----------------------
void Task2code( void * pvParameters ){
  Serial.print("Task2 running on core ");
  Serial.println(xPortGetCoreID());
  for(;;){
    if (!client.connected()) {
          reconnect();
          }
      client.loop();
    client.connect("ESP3_ambiente");
    now = millis();
    if (now - lastMeasure > 5000) {
      lastMeasure = now;
       h = dht.readHumidity();
       float bh;
       if(!isnan(h)){
          bh = h;
           Serial.println(h);
           dtostrf(h, 6, 2, humidityTemp);
           client.publish("huerto/ambiente/esp3/humedad", humidityTemp);
           }
             else{
            Serial.println(bh);
            dtostrf(bh, 6, 2, humidityTemp);
            client.publish("huerto/ambiente/esp3/humedad", humidityTemp);
            }
       t = dht.readTemperature();
       float bt;
       if(!isnan(t)){
          bt = t;

          Serial.println(t);
           dtostrf(t, 6, 2, temperatureTemp);
           client.publish("huerto/ambiente/esp3/temperatura", temperatureTemp);
           }
             else{
            Serial.println(bt);
            dtostrf(bt, 6, 2, temperatureTemp);
            client.publish("huerto/ambiente/esp3/temperatura", temperatureTemp);
            }
      Serial.print("Temperature: ");
      Serial.print(t);
      Serial.print(" *C  ");
      Serial.print("Humedad: ");
      Serial.print(h);
      Serial.print(" % \n  ");
    }
  }
}

void loop() {
}
String sendPhotoTelegram(){
  const char* myDomain = "api.telegram.org";
  String getAll = "";
  String getBody = "";

  camera_fb_t * fb = NULL;
  fb = esp_camera_fb_get();
  if(!fb) {
    Serial.println("Camera capture failed");
    delay(1000);
    ESP.restart();
    return "Camera capture failed";
  }
  Serial.println("Connect to " + String(myDomain));
  if (clientTCP.connect(myDomain, 443)) {
    Serial.println("Connection successful");
    String head = "--IrrigacionHuerto\r\nContent-Disposition: form-data; name=\"chat_id
```

Elaborado por: Estefania Acosta & Carlos Cuaical.



```
  chatId + "\r\n--IrrigacionHuerto\r\nContent-Disposition:form-data; name=\"photo\"; f
  String tail = "\r\n--IrrigacionHuerto--\r\n";
  uint16_t imageLen = fb->len;
  uint16_t extraLen = head.length() + tail.length();
  uint16_t totalLen = imageLen + extraLen;
  clientTCP.println("POST /bot"+BOTtoken+"/sendPhoto HTTP/1.1");
  clientTCP.println("Host: " + String(myDomain));
  clientTCP.println("Content-Length: " + String(totalLen));
  clientTCP.println("Content-Type: multipart/form-data; boundary=IrrigacionHuerto");
  clientTCP.println();
  clientTCP.print(head);
  uint8_t *fbBuf = fb->buf;
  size_t fbLen = fb->len;
  for (size_t n=0;n<fbLen;n=n+1024) {
    if (n+1024<fbLen) {
      clientTCP.write(fbBuf, 1024);
      fbBuf += 1024;
    }
    else if (fbLen%1024>0) {
      size_t remainder = fbLen%1024;
      clientTCP.write(fbBuf, remainder);
    }
  }
  clientTCP.print(tail);
  esp_camera_fb_return(fb);
  int waitTime = 10000;   // timeout 10 seconds
  long startTimer = millis();
  boolean state = false;
  while ((startTimer + waitTime) > millis()){
    Serial.print(".");
    delay(100);
    while (clientTCP.available()){
      char c = clientTCP.read();
      if (c == '\n'){
        if (getAll.length()==0) state=true;
        getAll = "";
      }
      else if (c != '\r'){
        getAll += String(c);
      }
      if (state==true){
        getBody += String(c);
      }
      startTimer = millis();
    }
    if (getBody.length()>0) break;
  }
  clientTCP.stop();
  Serial.println(getBody);
}
else {
  getBody="Connected to api.telegram.org failed.";
  Serial.println("Connected to api.telegram.org failed.");
}
return getBody;
}

void handleNewMessages(int numNewMessages){
  Serial.print("Handle New Messages: ");
  Serial.println(numNewMessages);

  for (int i = 0; i < numNewMessages; i++){
    // Chat id of the requester
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != chatId){
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }
    // Print the received message
    String text = bot.messages[i].text;
    Serial.println(text);

    String fromName = bot.messages[i].from_name;

    if (text == "/flash") {
      flashState = !flashState;
      digitalWrite(FLASH_LED_PIN, flashState);
    }
    if (text == "/photo") {
      sendPhoto = true;
      Serial.println("New photo request");
    }


    if (text == "/start"){
      String welcome = "Bienvenido al Huerto casero Telegram bot.\n";
      welcome += "/flash : Para encender el flash\n";
      welcome += "/photo : Para tomar una foto\n";
      bot.sendMessage(chatId, welcome, "Markdown");
    }
  }
}
```

Elaborado por: Estefania Acosta & Carlos Cuaical.

.

Anexo 27: Ingreso al fichero mymqtt.things.

```
root@openhabian:/home/openhabian# nano /etc/openhab/things/mymqtt.things
```

Fuente: Openhabian.

Anexo 28: Configuración de Bridge.

```
Bridge mqtt:broker:mymosquitto "Mosquitto" [ host="192.168.1.11", port=1883, secure=false, username="", password="", clientID="brokerClient" ]{
```

Fuente: Openhabian.

Anexo 29: Configuración de los Things.

```
Thing topic sensores "Humedad del suelo:" @ "Huerto" {
```

Configuración de Things en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 30: Configuración de canales.

```
Thing topic sensores "Humedad del suelo:" @ "Huerto" {
Channels:
Type string : estado2 "Modulo2" [qos=1,retained=true,stateTopic="huerto/sensores/esp2/lwt"]
Type number : voltaje "Voltaje" [ stateTopic="huerto/sensores/esp2/voltaje_sensor"]
Type number : voltaje1 "Voltaje1" [ stateTopic="huerto/sensores/esp2/voltaje1_sensor"]
Type number : voltaje2 "Voltaje2" [ stateTopic="huerto/sensores/esp2/voltaje2_sensor"]
Type number : porcentaje0 "Porcentaje0" [ stateTopic="huerto/sensores/esp2/porcentaje/porcentaje_0"]
Type number : porcentaje1 "Porcentaje1" [ stateTopic="huerto/sensores/esp2/porcentaje/porcentaje_1"]
Type number : porcentaje2 "Porcentaje2" [ stateTopic="huerto/sensores/esp2/porcentaje/porcentaje_2"]
Type number : volumen "Volumen0" [ stateTopic="huerto/sensores/esp2/volumen/volumen_agua_cont0"]
Type number : volumen1 "Volumen1" [ stateTopic="huerto/sensores/esp2/volumen/volumen_agua_cont1"]
Type number : volumen2 "Volumen2" [ stateTopic="huerto/sensores/esp2/volumen/volumen_agua_cont2"]
}
```

Configuración de canales en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 31: Creación del fichero Huerto.items.

```
root@openhabian:/home/openhabian# nano /etc/openhab/items/Huerto.items
```

Creación del fichero Huerto.items en Openhabian, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 32: Configuración para enlazar el canal.

```
String estado3_item "Modulo 3[MAP(transfor.map):%s]" <network> {channel="mqtt:topic:mymosquitto:ambiente:estado3"}
Switch lampara_item "Luz del Huerto" ["Lighting"] { channel="mqtt:topic:mymosquitto:ambiente:lampara" }
Number temperatura_item "Temperatura Interior [%.0f ºC]" <temperature> { channel="mqtt:topic:mymosquitto:ambiente:temperatura" }
Number humedad_item "Humedad Interior [%.0f %% ]" <humidity> { channel="mqtt:topic:mymosquitto:ambiente:humedad"}
Number setpoint3_item "setpoint 3 [%.0f %%]" <faucet>
```

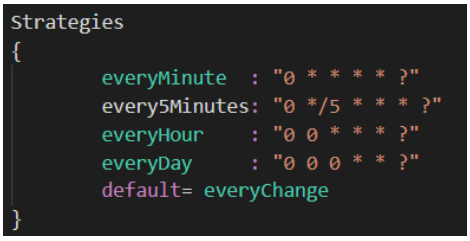
Configuración para enlazar el canal en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 33: Creación del fichero influxdb.persist.

```
root@openhabian:/home/openhabian# nano /etc/openhab/persistence/influxdb.persist
```

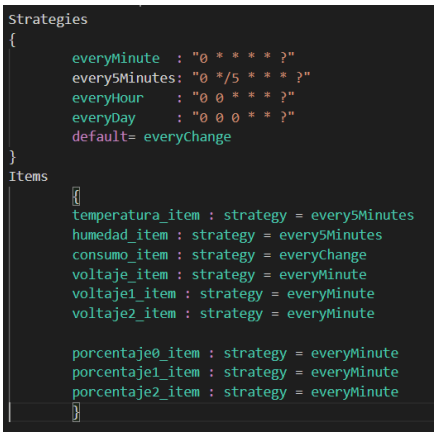
Creación del fichero influxdb.persit en Openhabian, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 34: Frecuencia para el almacenamiento de datos en InfluxDB.



Configuración de frecuencia de almacenamiento en InfluxDB realizado en Visual Studio Code Fuente: Openhabian.

Anexo 35: Elementos que se van a almacenar en la base de datos.



Configuración del fichero influxdb.persist en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 36: Creación de Interfaz Gráfica.

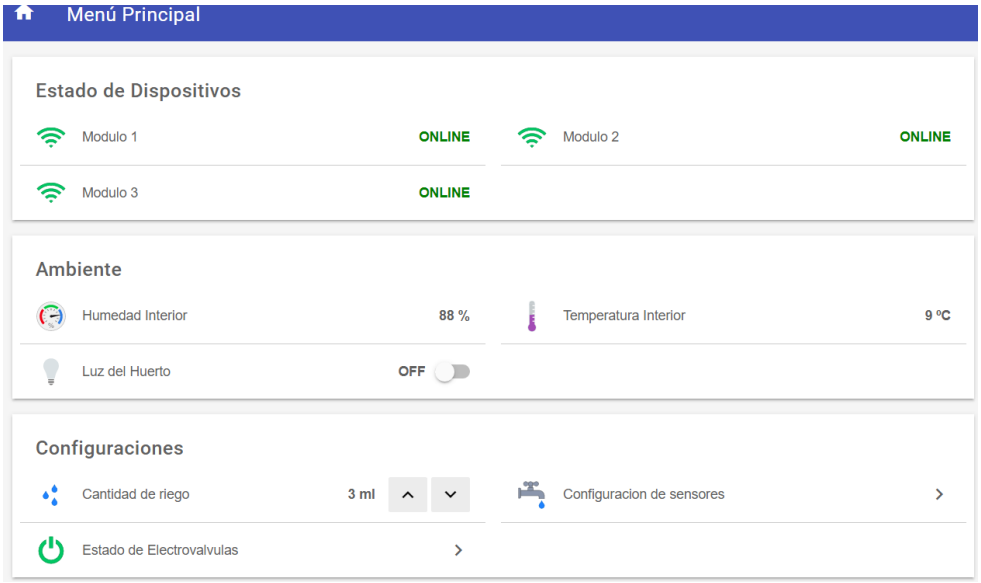
Creación del fichero panelbasico.sitemap en Openhabian, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 37: Configuración de panel básico.

```
sitemap panelbasico  label="Menú Principal" {
        Frame label="Estado de Dispositivos" {
        Text item=estado1_item icon="network" valuecolor=[ON="green",OFF="red"]
        Text item=estado2_item icon="network" valuecolor=[ON="green",OFF="red"]
        Text item=estado3_item icon="network" valuecolor=[ON="green",OFF="red"]
        }
        Frame label="Ambiente" {
            Text item=humedad_item
            Text item=temperatura_item
            Switch item=lampara_item icon="light"
           }
        Frame  label="Configuraciones"  {
                Setpoint item=setpoint_item  label="Cantidad de riego [%.0f ml]" icon="rain" minValue=20 maxValue=100 step=5
                Text  label="Configuracion de sensores" icon="faucet" {
                Setpoint item=setpoint1_item  label="porcentaje sensor 1 [%.0f %%]" icon="faucet" minValue=30 maxValue=90 step=1
                Setpoint item=setpoint2_item  label="Porcentaje sensor 2 [%.0f %%]" icon="faucet" minValue=30 maxValue=90 step=1
                Setpoint item=setpoint3_item  label="Porcentaje sensor 3 [%.0f %%]" icon="faucet" minValue=30 maxValue=90 step=1
                }
                        Text  label="Estado de Electrovalvulas" icon="switch" {
                                Switch item=valvula1_item icon="light"
                                Switch item=valvula2_item icon="light"
                                Switch item=valvula3_item icon="light"
                         }
        }
}
```

Configuración del fichero panelbasico.sitemap en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 38: Interfaz Gráfica en BasicUI.



Elaborado por: Estefania Acosta & Carlos Cuaical.

Anexo 39: Comando para fichero rules.

```
root@openhabian:/home/openhabian# nano /etc/openhab/rules/exam.rules
```

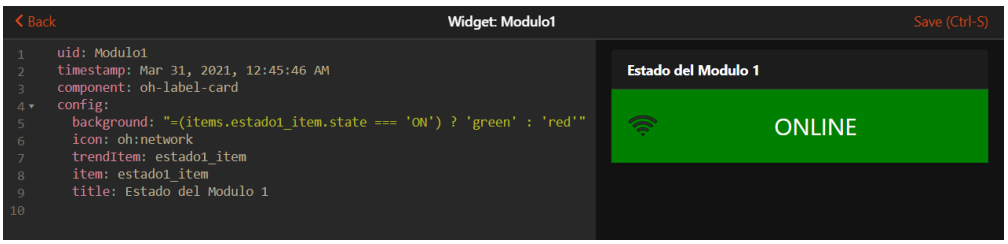
Creación del fichero exam.rules en Openhabian, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 40: Configuración para ejecutar una regla.

```
rule "comparacion"
when
    Item porcentaje0_item changed or Item setpoint1_item changed or Item porcentaje1_item changed or Item porcentaje2_item changed
then
 var i = 0
 while((i=i+1)<2){
 if   (setpoint1_item.state > porcentaje0_item.state){ sendCommand(caso1_item,"1"); } else { sendCommand(caso1_item,"0")}
 if   (setpoint2_item.state > porcentaje1_item.state){ sendCommand(caso2_item,"2"); } else { sendCommand(caso2_item,"0")}
 if   (setpoint3_item.state > porcentaje2_item.state){ sendCommand(caso3_item,"4"); } else { sendCommand(caso3_item,"0")}
 }
end
```

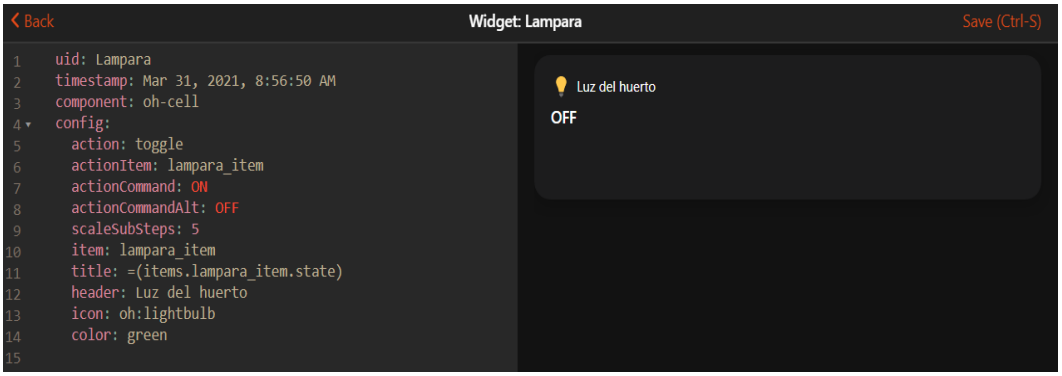
Configuración del fichero exam.rules en Visual Studio Code, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 41: Código YAML.



Configuración de Windget para visualizar la conexión de los dispositivos en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical

Anexo 42: Configuración de Widget.



Configuración de Windget para encender o apagar una luminaria en Openhab, Elaborado por: Estefania Acosta & Carlos Cuaical